**SANDIA REPORT**

SAND 2005-4330
Unlimited Release
Printed July 2005

# An Analysis of the Survivability of Sensor Darts in Impacts with Trees

John K. Prentice and David R. Gardner

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia National Laboratories

SAND 2005-4330
Unlimited Release
Printed July 2005

# An Analysis of the Survivability of Sensor Darts in Impacts with Trees

John K. Prentice
Sci-Tac, Inc.
505 Hapgood Street
Boulder, CO 80302-6965

and

David R. Gardner
Applied Computational Methods Department 9233
Sandia National Laboratories
Albuquerque, New Mexico 87185-0316

## Abstract

A methodology was developed for computing the probability that the sensor dart for the "Near Real-Time Site Characterization for Assured HDBT Defeat" Grand-Challenge LDRD project will survive deployment over a forested region. The probability can be decomposed into three approximately independent probabilities that account for forest coverage, branch density and the physics of an impact between the dart and a tree branch. The probability that a dart survives an impact with a tree branch was determined from the deflection induced by the impact. If a dart that was deflected so that it impacted the ground at an angle of attack exceeding a user-specified, threshold value, the dart was assumed to not survive the impact with the branch; otherwise it was assumed to have survived. A computer code was developed for calculating dart angle of attack at impact with the ground and a Monte Carlo scheme was used to calculate the probability distribution of a sensor dart surviving an impact with a branch as a function of branch radius, length, and height from the ground. Both an early prototype design and the current dart design were used in these studies. As a general rule of thumb, it we observed that for reasonably generic trees and for a threshold angle of attack of 5° (which is conservative for dart survival), the probability of reaching the ground with an angle of attack less than the threshold is on the order of 30% for the prototype dart design and 60% for the current dart design, though these numbers should be treated with some caution.

## Acknowledgement

The work reported here was funded by the “Near Real-Time Site Characterization for Assured HDBT Defeat” Grand-Challenge LDRD project.

# Table of Contents

## List of Figures

## List of Tables

Page

# An Analysis of the Survivability of Sensor Darts in Impacts with Trees

## 1 Introduction

In October 2003 Sandia started the “Near Real-Time Site Characterization for Assured HDBT Defeat” Grand-Challenge Laboratory-Directed Research and Development project with the vision to perform research and development on the key technologies for an active sensor system that will help assure the success of a nuclear earth-penetrating weapon strike on an underground strategic target through rapid site characterization and optimal aim-point selection. The objective of the characterization is to combine seismic imaging and detection results with penetrability measurements to locate homogeneous “soft spots” in the near surface of the site to improve the probability of weapon survival.

In the concept of operations a number of penetration probes and sensor darts are deployed from the air at a hard and deeply buried target (HDBT) site. The probes penetrate as deeply as possible and provide point penetrability measurements; they are subsequently detonated as seismic sources. The sensor darts lodge near the ground surface and collect the seismic signals; some also serve as communication relay nodes or as additional seismic sources. The relay nodes collect global-positioning system (GPS) position data, penetrability data, and seismic signals and transmit them via satellite or another overhead asset to a remote site for processing and fusing with existing intelligence from the site.

A sensor dart body has been designed and field tested in helicopter drop tests at Tonopah Test Range in both harder and softer targets. A typical dart design is shown in Figure 1. A concern is the survivability of the sensor dart in impacts with trees. The purpose of the study reported here is to assess the survivability of the sensor dart in such impacts.

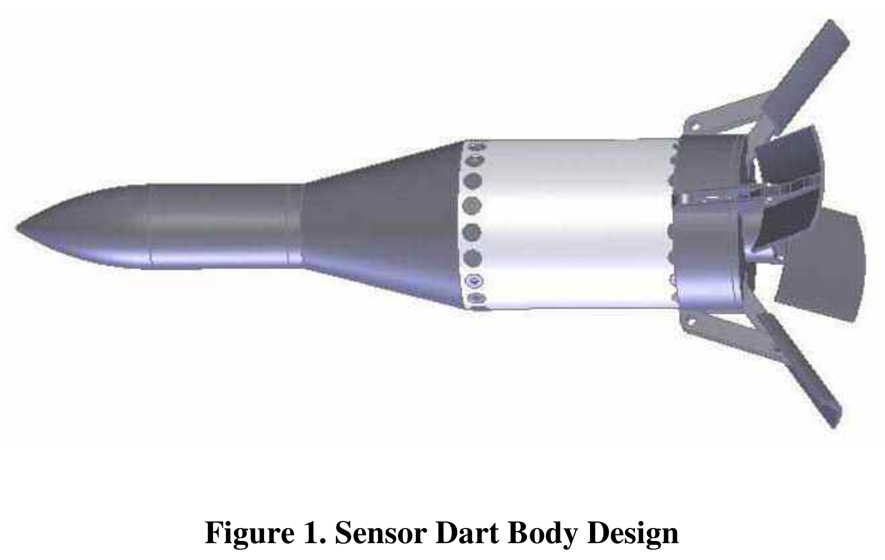

**Figure 1. Sensor Dart Body Design**

When the sensor dart is deployed over a forested region, some of the darts will impact tree branches prior to reaching the ground. While some of these impacts may physically damage the dart, others will simply perturb its trajectory or cause it to tumble. This latter issue is important because a relatively small angle of attack at ground impact may be sufficient to disable the dart. This report presents an analysis of the probability of survival of the sensor dart, taking into account impacts with trees, concentrating on the effect of glancing impacts that cause the dart to tumble.

The report is divided into several sections. Following this introduction, section 2 discusses the methodology for computing the probability of dart survival. Section 3 discusses available tools for determining the architecture of different kinds of trees. Section 4 gives some estimates for the survivability of an early prototype dart design if it hits a single branch assuming a range of tree characteristics. Section 5 gives some estimates for the survivability of the current dart design if it hits a single branch assuming a range of tree characteristics. Section 6 summarizes the report. The appendix consists of a listing of a Fortran 95 computer code for computing the probability of surviving the impact with a single tree branch, an important component of the calculation of the overall dart survival probability.

## 2 Calculation of the Total Survival Probability

When the sensor dart is deployed, two potential failure mechanisms are impact with a tree prior to reaching the ground or impacting a boulder or other hard object on the ground. This report addresses the tree-branch impact problem. Impact of the dart with a tree branch can lead to a deployment failure either by structurally damaging the dart or by imparting sufficient rotation so that it hits the ground with an excessive angle of attack and in the process breaks up or is damaged.

The total probability of dart deployment failure due to impact with a tree can be decomposed into three probabilities. The first is the probability $P_{canopy}$ of hitting the canopy of a tree. For a dart coming down normal to the ground, the canopy is defined as the total surface area of leaves and branches of the tree projected normal to the ground. The second factor $P_{branch}$ is the probability that, having come down on top of a tree, the dart will hit a branch in that tree (as opposed to the trunk, leaves, or small stems). The final factor is the probability $P_{impact}$ that the dart will *survive* the impact with a branch. Survival is based on the dart either not being damaged or not having an excessive rotation at ground impact. To a reasonable approximation, these probabilities can be treated as independent, in which case the probability that the dart will *survive* deployment is

$$P_{survival} = (1 - P_{canopy}) + P_{canopy}(1 - P_{branch}) + P_{canopy}P_{branch}P_{impact},$$

or more simply,

$$P_{survival} = 1 - P_{canopy}P_{branch}(1 - P_{impact}). \tag{1}$$

The first probability $P_{canopy}$ is arguably the easiest to determine. This is nothing more than the fraction of the total impact surface area that is covered by the tree canopy when viewed from above (normal to the ground),

$$P_{canopy} = \frac{A_{total\,canopy}}{A_{total}} \tag{2}$$

where $A_{total}$ is the total surface area of the impact region and $A_{total\,canopy}$ is the total area covered by tree canopy. This can be assessed by aerial or satellite remote sensing in the visible spectrum. Note that heavily forested areas, areas in which $P_{canopy}$ is near 1, are unlikely target sites because earth penetrators, like the sensor darts, are subject to deflection by impacts with trees.

The second term, $P_{branch}$, is

$$P_{branch} = \frac{A_{branch}}{A_{canopy}} \tag{3}$$

where $A_{branch}$ is the total area of branches within a single tree canopy projected normal to the ground and $A_{canopy}$ is the projected area of the canopy for that tree[1]. Strictly speaking, this should be a probability density distribution and the calculation should be an integral

[1] This area is given approximately by the area of a circle normal to the vertical with a diameter equal to the widest point on the tree, including all leaves, branches, and stems.

over that distribution[2]. However, if the region of interest is covered by trees of largely the same species and maturity, it may be sufficient to simply choose an average branch area density and use Eq. (3) as an approximation. In either case, the practical difficulty is computing the branch areal density. For deciduous trees, it may be possible to determine this density during winter months using aerial or satellite remote sensing. Alternately, it may be determined based on knowledge of the tree species and maturity using various growth models (see section 3).

The final term, $P_{impact}$, requires an analysis of the dynamics of the sensor dart during the impact with a tree branch. This is the only part of the analysis that is dependent on the specific characteristics of the sensor dart or requires any physics-based analysis. As with the previous term, this is strictly speaking a probability distribution, not a single number. However, as will be discussed in section 4, numerical experiments suggest that it may be possible to assign a single value to this probability that is sufficient for most purposes.

We devote the remainder of this section to a discussion of a methodology for calculating the single branch impact survivability probability $P_{impact}$. For the purposes of this analysis, the sensor dart is considered to be a rigid body of rotation and the tree branch is considered to be a statically determinant, perfectly elastic beam. During an impact, the dart causes the branch to bend and the resulting elastic restoring force generates an impulse on the dart that changes the dart center of mass velocity and the angular velocity around the center of mass. The main approximations in this methodology are that the dart is treated as a non-deformable rigid body and aerodynamic forces are neglected. The rigid body approximation is reasonable because impacts capable of deforming or damaging the projectile are expected to induce a pitch and yaw to the rigid body that would result in destruction at ground impact irrespective of damage done during the branch impact. The neglect of aerodynamic forces is also considered acceptable given that any impact with a tree will occur close to the ground.

Two coordinate systems are used in modeling the dynamics of the dart. The first is an inertial, or laboratory, coordinate system fixed in space. The location of the origin and orientation of the axes of this system are arbitrary so long as they are fixed in space. The second coordinate system is the body coordinate system. The sensor dart is assumed to be a body of revolution. The origin of the body coordinate system is fixed at the center of mass of the dart with the positive $z$ axis pointing toward the nose of the dart along its centerline. The other two axes are perpendicular to the centerline, but otherwise their orientation is arbitrary since the dart is a body of revolution.

We now derive the equations of motion for the sensor dart. The dart is a non-deformable rigid body with moment of inertia $\overset{\llcorner}{I}$ given by

$$I_{ik} = \int dV \rho \left( x_i^2 \delta_{ik} - x_i x_k \right) \tag{4}$$

[2] The finite projected area of the dart can be neglected for trees that are of significant size compared to the dart. A tree for which $A_{canopy}$ is of comparable scale with the dart is unlikely to deflect the dart significantly.

where $\rho$ is the material density, $x_i$ is the $i$th material coordinate in the dart body coordinate system, $\delta_{ik}$ is the Kronecker delta function, and the integration is over the volume of the dart[3]. The mass of the dart is $\mu$.

The equations of motion of the center of mass of the dart are

$$\mu \frac{d\vec{r}_{cm}}{dt} = \sum \vec{F}_i \,, \tag{5}$$

where $\vec{r}_{cm}$ is the radius vector of the center of mass in the laboratory coordinate system and $\vec{F}_i$ are the external forces acting on the dart during the time step. One of these forces is gravity[4], the other forces are those exerted by the branch on the dart.

The equations of rotational motion for the dart are given in the dart body coordinate system by Euler's equations [1]:

$$I_x \frac{d\Omega_x}{dt} + (I_z - I_y)\Omega_y\Omega_z = \tau_x \tag{6}$$

$$I_y \frac{d\Omega_y}{dt} + (I_x - I_z)\Omega_z\Omega_x = \tau_y \tag{7}$$

$$I_z \frac{d\Omega_z}{dt} + (I_y - I_x)\Omega_x\Omega_y = \tau_z \tag{8}$$

where $\vec{\Omega}$ is the angular velocity of the dart in the dart body coordinate system, $\vec{\tau}$ are the external torques exerted on the dart due to the branch in the dart body coordinate system, and $I_x$, $I_y$, and $I_z$ are the principal moments of inertia. Note that these principal moments of inertia are assumed to be parallel to the *x*, *y*, and *z* axes respectively of the dart-body coordinate system.

To calculate the torques $\vec{\tau}$ in the dart body coordinate system, we begin with the torques $\vec{T}$ expressed in the laboratory coordinate system. Assume that the forces $\vec{F}_i$ are exerted at points $\vec{R}_i$ on the surface of the dart where the vectors $\vec{R}_i$ are in the laboratory coordinate system. Relative to the center of mass in the laboratory coordinate system, the external torques on the dart are then

$$\vec{T} = \sum \left( \vec{R}_i - \vec{r}_{cm} \right) \times \vec{F}_i \,. \tag{9}$$

The external torques $\vec{\tau}$ in the dart body coordinate system are related to the torques $\vec{T}$ expressed in the laboratory coordinate system by the transformation

$$\vec{\tau} = E\vec{T} \tag{10}$$

[3] The moment of inertia for the sensor dart will in general be calculated by the CAD package used in the design of the dart.

[4] However, as a practical matter, gravity can be neglected given the velocity of the dart and its proximity to the ground during any branch impact.

where the rotation tensor $E$ is given by [2]

$$E = \begin{pmatrix} \cos\psi\cos\phi - \cos\theta\sin\phi\sin\psi & \cos\psi\sin\phi + \cos\theta\cos\phi\sin\psi & \sin\psi\sin\theta \\ -\sin\psi\cos\phi - \cos\theta\sin\phi\cos\psi & -\sin\psi\sin\phi + \cos\theta\cos\phi\cos\psi & \cos\psi\sin\theta \\ \sin\theta\sin\phi & -\sin\theta\cos\phi & \cos\theta \end{pmatrix} \quad (11)$$

where $\phi$, $\theta$, and $\psi$ are the Euler angles of the dart body coordinate system relative to the laboratory coordinate system. An equivalent rotation matrix can be written in terms of the direction cosines between the two coordinate systems. Note that the calculation of the torque can also be done directly in the body coordinate system, thereby eliminating the need for the rotation in Eq. (10), though it may in this case instead be necessary to transform some description of the branch from the laboratory to body coordinate systems.

The time rate of change of the angular velocity of the sensor dart in the body coordinate system can be expressed in terms of the time rate of change of the Euler angles in the laboratory coordinate system, thereby producing a set of differential equations describing the dart rotation in the laboratory system. However, this requires a complicated inversion of three simultaneous differential equations in order to express the rate of change of the Euler angles in terms of the body angular velocity vector. A less complex solution can be obtained by noting that over small time intervals, the impact induced torques produce an infinitesimal rotation of the dart around the instantaneous dart angular velocity vector $\vec{\Omega}$. This has the effect of taking a point $\vec{r}_o$ on the dart in the body coordinate system into a new point $\vec{r}_n$ where [3]

$$\vec{r}_n = \vec{r}_o \cos\phi + \hat{n}\left(\hat{n}\cdot\vec{r}_o\right)\left[1-\cos\phi\right] + \left(\hat{n}\times\vec{r}_o\right)\sin\varphi. \quad (12)$$

The vector $\hat{n}$ is a unit vector pointing along the angular velocity vector in the body coordinate system,

$$\vec{\Omega} = \hat{n}\frac{d\phi}{dt}, \quad (13)$$

and the value $\phi$ in equation (12) is equal to $\frac{d\phi}{dt}\cdot\Delta t$. These new coordinates are then rotated back into the laboratory coordinate system using the transpose of Eq. (11) to give the orientation of the dart in the laboratory coordinate system at the end of this small time step.

This suggests a numerical scheme for integrating the equations of motion of the dart during the impact with a branch. We integrate the equations over small time steps $\Delta t$. At the beginning of a time step, we transform the coordinates of the tree and the angular velocity vector from the laboratory coordinate system to the body coordinate system using the rotation matrix in Eq. (11) or its equivalent in terms of direction cosines. We then calculate the force on the sensor dart due to the tree using elastic beam theory in the body coordinate system. Next we calculate the infinitesimal rotation of the principal axes of the sensor dart in the body coordinate system due to the resulting torques on the dart. Upon rotation back into the laboratory coordinate system, these axes represent the new orientation of the body coordinate system relative to the laboratory coordinate system from which new Euler angles (or equivalently, direction cosines) can be calculated. The center of mass motion of the dart is calculated by integrating Eq. (5) over the time step. A second-order accurate numerical integration is used for both the rotations and center of mass motion.

This method for calculating the sensor dart motion is exact within the approximation introduced by the numerical quadrature. Numerical experiments suggest that a few hundred time steps are sufficient over the duration of the dart/branch impact to achieve good numerical accuracy. This implies time steps on the order of tens of microseconds.

A more difficult problem is calculating the dynamics of the tree due to the impact with the dart. An exact treatment of this problem would require solving the differential equations for the elastic motion of the tree during the impact. These equations include the motion of elastic waves which travel at approximately 1 km/s along the axis of the tree branch. A numerical solution of these elastic equations requires a time step minimally one order of magnitude smaller than that required for a *very* fine resolution of the dart motion. While this is not an unreasonable computational burden for a single impact calculation, the calculation of the impact probability $P_{impact}$ can require hundreds of thousands of Monte Carlo dart/branch calculations for different impact initial conditions. This makes it desirable to use the largest time step possible in order to make the problem computationally feasible. In addition, a detailed calculation of the tree dynamics is probably not necessary in order to estimate the effect of an impact on the dart given the uncertainties associated with tree architecture (see the next section), provided that the approximate dart torque and center of mass forces can be calculated to a reasonable approximation.

To that end, the scheme used in this analysis is as follows. The unbent tree branch is defined by the location of its two endpoints. The end where the branch connects to the trunk is fixed in space while the other end is allowed to move. At the beginning of a simulation, the dart is positioned with its axis aligned vertically and the nose just touching some portion of a tree branch. The calculation is done such that the deflection of the branch will always occur in the same half-space relative to the axis of the dart. This simply imposes the physical constraint that the branch always deflects to one side or the other of the dart. During a time step, the dart will move some distance. If it is in contact with the branch, this will result in some small additional deflection of the branch. Knowing the contact point between the branch and the dart at the end of this time step, and hence the deflection of the branch, the force on the dart at the beginning of the next time step can be calculated using elastic beam theory [4] as

$$F = \frac{3\xi I_t \delta}{L^3}, \tag{14}$$

where $\xi$ is the modulus of elasticity (Young's modulus) of the tree, $I_t$ is the moment of inertia of the tree branch cross-section (assumed here to be a circle of some radius), $\delta$ is the deflection of the branch at the impact point, and $L$ is the distance from the tree trunk to the impact point. Note that this is a statically determinant solution that neglects all momentum effects.

The difficulty is that in the absence of solving the equations for the true motion of the branch, the contact point between the dart and the branch is not known. However, it can be estimated using the heuristic that the branch will be in contact with the point on the dart where the deflection is the smallest. This is consistent with a minimum potential energy solution for beam deflection, but it assumes the branch can move fast enough to always satisfy this requirement. To compute the point on the dart where the deflection will be smallest, multiple points along the dart from the nose to the tail are sampled. For each of these points, the deflection of the tree is calculated such that the tree is always on the same side of the dart. This calculation is done assuming the correct exterior geometry of the dart and modeling the tree as a cylinder of prescribed length and radius fixed at the end where the branch intersects the trunk of the tree. The point where the minimum deflection occurs is assumed to be the impact point.

To calculate the impact probability $P_{impact}$, a Monte Carlo approach is used. Ranges of heights, radius, and lengths are defined for branches representative of a particular species of tree. A specific branch height, radius, and impact point (less than or equal to the branch length) are chosen at random from within these ranges. A calculation of the trajectory of the sensor dart is then performed. If the dart experiences a rotation that exceeds some threshold it is assumed to not have survived the impact. Otherwise it will be assumed to have survived. This is repeated $N_{total}$ times, where $N_{total}$ is usually on the order of 50,000 to 100,000 with new randomly chosen impact parameters for each calculation. The results of these calculations can then be used to define a probability distribution in terms of the branch height, radius, and length. Alternately, a single average probability of survival of the dart for these conditions can be defined by

$$P_{impact} = \frac{N_{survive}}{N_{total}} \tag{15}$$

where $N_{survive}$ is the total number of calculations where the dart impacted the ground with an angle of attack less than or equal to the maximum survivable angle. A computer code written in Fortran 95 for calculating the average probability is given in the appendix.

Note that in some cases, the dart could impact more than one branch during its descent through a tree. The computer code given in the appendix and the analyses in sections 4 and 5 neglect this. The modification to the computer code to include multiple branch impacts is relatively trivial. However, it is probably not necessary in general to include the effect of multiple branch impacts. As discussed in sections 4 and 5, numerical experiments suggest that a single impact with a larger branch generally induces sufficient tumbling of the dart so that it will not survive impact. On the other hand, impact with very small branches generally has minimal effect on the dart, making multiple such impacts of little concern and neglecting them is not believed to represent a significant error.

## 3 Tree Architecture and Predictive Tools

The calculation of both $P_{branch}$ and $P_{impact}$ require some knowledge of the architecture of the trees present in the impact zone. In the case of $P_{branch}$, this information is important in determining either the probability distribution of or an average for the size and density of branches in the trees in order to determine the total projected branch area. In the case of $P_{impact}$, it is necessary to know something of the probability distribution for the height of the trees in the impact region, their branch radii, and branch lengths. This information is then used in the computer code discussed in the appendix to compute the probability of survival of the dart during a branch impact using the methodology outlined in the previous section.

Ideally, the architecture of the trees requires either direct sampling via remote sensing or on-the-ground inspection or knowledge of the general characteristics of similar trees growing under similar climatic and soil conditions. A detailed discussion of this topic is beyond the scope of this paper or the competence of the author. However, it may be instructive to offer some general comments about tree architecture and review some tools that are available for modeling tree characteristics.

The maturation of a tree involves two growth processes: the forming of shoots of the crown and the roots (primary growth) and expansion of the stem and root diameter (secondary growth) [5]. A combination of these processes, following an architectural model common to each species and influenced by environmental factors (climate and soil in particular), gives each species a characteristic aerial and subsurface structure and form. Broadly speaking, trees have one of two forms. Excurrent tree forms, typical of northern conifers and a few deciduous trees such as sweetgum and tuliptree, grow in a conical form that is the result of selection pressures of snow, ice, and wind. These trees tend to have a single trunk with smaller branches growing radially out from it. However, not all conifers exhibit this growth pattern; some change to a more rounded and spreading form related to climate.

The second common form is called decurrent or deliquescent. These trees have lateral branches that grow nearly as fast as or faster than the trunk. The trunk may fork repeatedly, giving rise to a spreading form. Examples of such trees are elms, oaks, maples, and other species that grow in less harsh environments than is typical of those where excurrent trees are usually found.

Within these two broad categories, it is possible to reduce the diversity of tree forms to 23 architectural models. However, a given species may be intermediate between two models or share features of several models. As discussed in Barnes, *et al*., “[c]riteria for identifying the models included whether shoot extension is rhythmic or continuous, whether the tree is unbranched (as in some palms) or branched, whether or not branch differentiation is orthotropic (erect) or plagiotropic (horizontal), and whether flowering is terminal or lateral”. Good review papers discussing tree architecture are those of Tomlinson [6] and Barthelemy, *et al*. [7].

An important characteristic of tree architecture is reiteration. When a tree is damaged and new shoots arise, these shoots repeat the original architectural pattern of the species. A common example of this occurs when a trunk is pruned or cut and shoots looking like new versions of the same kind of tree grow out of the old trunk. Reiteration can lead to the development of multiple branches and trunks, even in excurrent trees such as conifers.

The canopy of a tree consists of the branches, stems, and leaves. More generally, it refers to the sum of all of the individual canopies of what can be multiple trees and shrubs at several layers in the undercanopy. Note that the presence of multiple layers within a canopy could complicate the analysis of the dart survivability since in some ecosystems there may be layers of trees of different species with very different architectural characteristics. In such cases, it will probably be necessary to take into account the possibility of impact with more than one tree in the calculation of the impact probability $P_{impact}$.

Despite being able to reduce tree architectures into only 23 models, the diversity of tree species is still dizzying[5]. Adding to this is the range of variability in the size of trees. In Missouri, tulip poplar in the Bootheel region can reach over 50 meters, while the shrub-like wild plum tops out at around 4 meters in a north-central Missouri fencerow [8]. In Ontario, canopy trees range from 20 to 25 meters in height. Some giants may reach heights of 25 to 30 meters or more. Beneath the canopy there is another layer of vegetation, call the subcanopy or understory. Trees forming in this layer typically reach heights of 7 to 10 meters. Beneath this layer is the shrub layer, formed by shrubby species such as viburnums, which grow in heights to perhaps 3 meters [9]. Even more impressive, in tropical regions the trees in the upper level of the canopy can reach to heights exceeding 65 meters with an understory that stretches for vast distances [10].

Interestingly, even in heavily forested regions, trees rarely form an unbroken canopy over large areas. Even in tropical rainforests, branches will overlap but rarely interlock or even touch. Instead they are separated by a few feet. It is not known why this happens, but one theory is that it might serve as protection from infestations from tree-eating caterpillars and tree diseases [10]. This is of obvious importance to the sensor dart survivability since it implies than even in densely forested regions, there will be gaps between trees where a dart is unlikely to impact a branch. Similarly, even within the canopy of a single tree, the trunk and major branches often occlude only a fraction of the total projected surface area of the tree. For example, though constituting an unusually sparse tree, the total projected surface area of the stems and branches in walnut trees amounts to only approximately 1/5 of the projected canopy surface area [11]. This means that $P_{branch} \sim 0.2$ for walnut trees. In this case, even if both $P_{canopy}$ and $P_{impact}$ were equal to one (implying 100% of the ground was covered by overlapping trees and any impact with a branch would disable the sensor dart), the probability of the dart surviving to impact would still be 80%. However,

[5] A sense of this diversity can be gained by looking at the MUC Dictionary at http://ael.physic.ut.ee/kaariku/mucdictionary.htm (11 April 2005) or the CWHR Classification System at http://frap.cdf.ca.gov/projects/frap_veg_frames/classification.html (11 April 2005).

this is unquestionably an upper bound; most species of trees exhibit a much higher projected branch surface area. In the case of some conifers, eyeball estimates suggest this number could reach 80%, meaning $P_{branch} \sim 0.8$, implying a rather low probability of the dart surviving passage through such a tree.

Though cursory, this short discussion of tree architecture illustrates the immense diversity of trees and the impossibility of drawing general conclusions about the probability of dart survival without reference to the nature of the trees growing in a specific impact zone. It will be essential to characterize the nature of the trees in each impact zone in order to derive any meaningful information about $P_{canopy}$, $P_{branch}$, or $P_{impact}$.

Some remote sensing techniques exist that can be employed to characterize tree architecture in a given region. For example, total canopy coverage for use in calculating $P_{canopy}$ can be obtained by aerial or satellite remote sensing in the visible spectrum. A technique discussed in the open literature for measuring canopy height and foliage density, of value in determining both $P_{branch}$ and $P_{impact}$, is aerial pulsed infrared laser profiling (lidar) [12, 13]. Other, more sophisticated, techniques no doubt exist.

Computer modeling is another tool that can be used to estimate $P_{branch}$ and $P_{impact}$. There has been extensive work done in recent years to develop computer models for predicting the growth and architecture of trees based on species and growing conditions. We will briefly describe some of these models; however, this discussion is by no means exhaustive. We begin by discussing models that incorporate biological information to predict growth or structure.

An early model for describing tree architecture of densely wooded conifer forests was developed by Sandia/California in the 1980s and documented in a laboratory report [14]. This model used algebraic equations to compute canopy coverage, crown height, trunk and branch diameters, and tree density based on soil and rainfall parameters, forest density parameters, and the age of the forest.

The FORECAST model is an "ecosystem-based, stand-level, forest growth simulator"[6]. This model incorporates information on decomposition, nutrient cycling, light competition, and other ecosystem properties to simulate forest growth and ecosystem dynamics. This model would appear to be most useful in estimating the total canopy cover over a conifer forest and may also be useful in inferring information about branch density and size within the trees in that forest.

The LIGNUM model is a "functional-structural model that represents a tree using four modeling units which closely resemble the real structure of trees: tree segments, tree axes, branching points and buds. Metabolic processes are explicitly related to the structural units in which they take place" [15]. This model has been successfully applied to broad-leaf, deciduous trees.

[6] FORECAST was developed and is available from Clive Welham, Brad Seely, and Hamish Kimmins, Forest Ecosystem Management Simulation Group, University of British Columbia, Vancouver, BC.

So-called pipe model theory can be used to predict the height growth of trees [16] and the branch distribution [17] based on a metabolic model of height growth and site index derived from a parameterization of the annual carbon balance of a tree.

In addition to these biologically based models, there has been extensive development of "topological" models that are not based on biological, first-principle, growth models but have nonetheless been successfully applied to modeling tree architecture. These models may also be of value in estimating $P_{branch}$ and $P_{impact}$. However, the non-biological nature of the model will presumably limit their utility to previously well-characterized species. We will conclude this section with a brief overview of some of these models.

Multiscale Tree Graph (MTG) is a topological model suitable for representing tree topology with respect to scale and time. It is capable of expressing succession and branching relationships between plant components [18, 19, 20].

L-systems are another non-growth based model capable of representing tree architecture. The approach involves several steps:

1. "Individual-based simulation of a plant ecosystem, using coarse representations of individual plants. This step establishes positions and general characteristics of the plants, such as their height, diameter, and the overall shape of tree crowns.
2. Generation of detailed plant models that satisfy the characteristics determined in the previous step.
3. Realistic visualization of the ecosystem, using detailed plant models substituted for their coarse counterparts."

The resulting "formalism provides a tool for specifying individual-based ecosystem models and simulating their development over time" [21, 22].

The AMPmod model describes tree architecture using both topological and geometric information and "is based on the use of a multiscape model of plant topology—called multiscale tree graphs—which is extended to include geometry." The "method … does not depend on the plant species or on the geometric model used to describe the plant components" [23, 24]. It has been successfully applied to model apple and walnut trees. Two additional generic models (largely computer-graphics oriented) are those by Yan, *et al*. [25], and Bloomenthal [26].

# 4 Calculations of $P_{impact}$ for a Prototype Dart Design

As discussed in section 3, the diversity of tree architecture makes it impossible to draw detailed conclusions about the survivability of the sensor dart without reference to the characteristics of a specific impact site. In particular, estimates of $P_{canopy}$ and $P_{branch}$ will inevitably be very site-specific. However, we can make some general comments regarding $P_{impact}$ , the probability of survival of an impact with a single branch, by considering calculations of $P_{impact}$ for a range of elastic moduli for green wood.

In this section we consider calculations of the probability $P_{impact}$ that a prototype sensor dart will reach the ground with a survivable angle of attack. The calculations were conducted with the computer code described in section 2 and given in the appendix. The geometry, mass, and moment of inertia of the dart were supplied by Robert T. Gilchrist.

## 4.1 Modeling Assumptions

We assumed the dart was delivered with its design velocity of 75 m/s normal to the ground. Note that for this study the dart terrabrakes were assumed to be closed during the impact with the branch.

We assumed the threshold survival angle of attack for the prototype sensor dart is 5°. This value is conservative value for dart survival: in drop tests at Tonopah Test Range the dart impacted the ground at angles of attack significantly larger than this and the dart embedded in the ground and its internal components survived [27].

We assumed a uniform probability distribution for each of the various branch parameters. The probability of survival was calculated using the code given in the appendix, which permits the height, diameter, and distance from the trunk to the impact point on the branch to be varied within some specified range. A Monte Carlo calculation was then performed, in each iteration of which a specific branch height, a diameter, and a distance from the trunk to the impact point were randomly selected. The average single branch impact survival probability $P_{impact}$ is calculated using Eq. (15).

We assumed there is no correlation between the radius of the branch and the distance of the impact point from the branch. In general, the greater the distance between the trunk and the impact point, the smaller the branch radius will be.

Finally, we assumed there is no correlation between the branch height and the branch radius. In general, branches at a greater height will tend to have a smaller radius at a given impact distance from the trunk than branches closer to the ground. Thus the values calculated for $P_{impact}$ are likely under-estimates of the probabilities.

## 4.2 Moduli of Elasticity for Green Woods

Three sets of calculations were conducted. Each set used a wood with a different modulus of elasticity: one with a low modulus, one with a high modulus, and one with an intermediate modulus (Table 1).

The modulus of elasticity of wood varies with a variety of factors, including growing conditions, environmental and insect damage, age, and moisture content. For example, the modulus of elasticity for Scots pine (*Pinus sylvestris* L.) was approximately four times greater for trees 25 years old than for trees 7 years old and the modulus of elasticity varied significantly for moisture contents less than 30% [28]. For this study we use the properties of clear green wood specimens, taken from Green, *et al*. [29].

For a wood with a low modulus of elasticity, we used ceiba (*Ceiba pentandra*), also known as the silk-cotton tree and the kapok tree. Ceiba is distributed worldwide in the tropics. It is found from the Tropic of Cancer in Mexico southward through Central America to Columbia, Venezuela, Ecuador, and Brazil; in the Malay Peninsula; and in West Africa. The wood is used commercially for plywood, packaging, light construction, and pulp and paper products. The green wood has a modulus of elasticity of 2.80 GPa (moisture content of 73%) [29, 30, 32, 33].

**Table 1. Moduli of Elasticity for Green Woods**

| **Wood** | **Modulus of Elasticity (Green Wood) [29]** |
|---|---|
| Ceiba (*Ceiba pentandra*) | 2.80 GPa |
| White Oak (*Quercus alba*) | 8.60 GPa |
| Kaneelhart (*Licaria* spp.) | 26.3 GPa |

For a wood with a high modulus of elasticity, we used kaneelhart (*Licaria* spp.), also known as the brown silverballi. Kaneelhart is found mostly in the Guianas. The wood is used commercially for furniture, boat building, heavy construction, and parquet flooring. The green wood has a modulus of elasticity of 26.3 GPa (at a moisture content of 73%) [29, 30, 31].

For a wood with an intermediate modulus of elasticity, we used white oak (*Quercus* alba), also known as American white oak and Arizona oak. White oak is widely distributed throughout the United States. The wood is used commercially for ships, railroad ties, timber bridges, fuel wood, flooring, furniture, veneer, plywood, kegs and casks, pallets, caskets, boxes, and paneling. The green wood has a modulus of elasticity of 8.60 GPa (at a moisture content of 73%) [29, 34].

## 4.3 Probability Convergence Study

To verify the convergence of the probability calculations, we varied the number of calculations per bin, $N_c$, from 100 to 1000 for the average probability of survival. 62,500 Monte Carlo samples were sufficient to achieve numerical convergence (Table 2).

**Table 2. Convergence of the Average Survival Probability $P_{impact}$ with the Number of Calculations per Bin, $N_c$, for the Prototype Sensor Dart Design**

|  | Average Survival Probability |  |  |
|---|---|---|---|
| $N_c$ | Ceiba | White Oak | Kaneelhart |
| 100 | 0.5025 | 0.4140 | 0.3406 |
| 200 | 0.4999 | 0.4124 | 0.3415 |
| 250 | 0.4998 | 0.4126 | 0.3420 |
| 300 | 0.4988 | 0.4107 | 0.3405 |
| 400 | 0.5012 | 0.4137 | 0.3439 |
| 500 | 0.4992 | 0.4121 | 0.3418 |
| 1000 | 0.5002 | 0.4132 | 0.3421 |

## 4.4 Survival Probability as a Function of Branch Impact Location

First we consider the sensitivity of the average impact survival probability to how close the impact point on a branch is to the trunk of a tree[7]. In these calculations, the tree branch radius varied randomly between 0.5 cm and 10 cm and the height of the tree varied randomly between 1 m and 30 m (approximately 100 ft). The distance between the trunk and the impact point was constrained to ranges of 25 cm. The calculated average impact survivability probabilities are plotted in Figure 2 and values are given in Table 3. Recall that the survival probability given here is only $P_{impact}$; it is not the overall survival probability for the dart deployed over a forest. Calculation of the overall survival probability requires knowledge of all three probabilities in Eq. (1).

The survival probability decreases as the impact point gets closer to the trunk for each of the tree species. The physical explanation is obvious: the closer to the trunk the impact occurs, the stiffer the branch will be. Note that the sensor dart radius at its widest point is 14 cm (5.5 inches). An impact closer than this distance to the trunk would involve actually hitting the trunk, which would in general be unsurvivable for most trees. Not surprisingly, the probability of survival varies with the elastic modulus of the wood: The larger the elastic modulus of the wood is, the lower the average survival probability.

[7] Note that the distance between the trunk and the impact point is statistically related to the length of the branch since distances within some range along a particular branch are sampled.

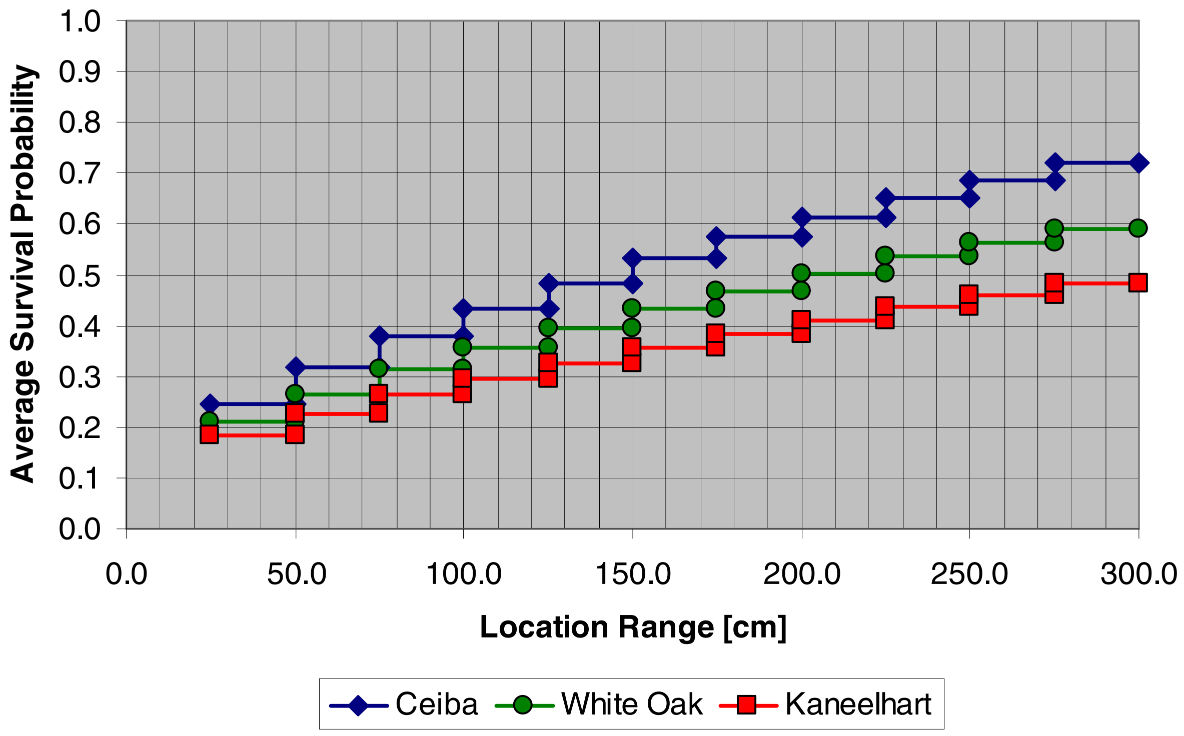

**Figure 2. Average Survival Probability, $P_{impact}$, for a Prototype Dart Design as a Function of Impact Location Measured from the Tree Trunk, $l$**

**Table 3. Average Survival Probability $P_{impact}$ for the Prototype Dart Design as a Function of Impact Location Measured from the Tree Trunk, $l$**

| | | $P_{impact}$ | |
|---|---|---|---|
| **$l$ Range (cm)** | **Ceiba** | **White Oak** | **Kaneelhart** |
| 25.0 – 50.0 | 0.244 ± 0.002 | 0.210 ± 0.002 | 0.184 ± 0.002 |
| 50.0 – 75.0 | 0.316 | 0.266 | 0.226 |
| 75.0 – 100.0 | 0.378 | 0.312 | 0.263 |
| 100.0 – 125.0 | 0.433 | 0.356 | 0.295 |
| 125.0 – 150.0 | 0.485 | 0.396 | 0.328 |
| 150.0 – 175.0 | 0.532 | 0.434 | 0.356 |
| 175.0 – 200.0 | 0.574 | 0.468 | 0.384 |
| 200.0 – 225.0 | 0.613 | 0.502 | 0.411 |
| 225.0 – 250.0 | 0.651 | 0.534 | 0.436 |
| 250.0 – 275.0 | 0.686 | 0.564 | 0.460 |
| 275.0 – 300.0 | 0.719 | 0.591 | 0.484 |

## *4.5 Survival Probability as a Function of Branch Radius*

Next we consider the sensitivity of the branch impact survival probability to the radius of the branch. In these calculations, the distance of the impact point from the trunk varied randomly between 25 cm and 300 cm and the height of the tree varied randomly between 30 cm and 30 m (approximately 100 ft). The radius of the branch was constrained to relatively narrow ranges of 0.5 cm.  The calculated average impact survivability probabilities are plotted in Figure 3 and values are given in Table 4.

The survival probability decreases rapidly as the branch radius increases (Figure 3 and Table 4). This is a consequence of the fact that the elastic restoring force of the branch is proportional to the area moment of inertia of the branch which in turn is proportional to the fourth power of the radius of the branch. As before, the higher the elastic modulus of the wood is, the lower the probability of survival.

## *4.6 Survival Probability as a Function of Branch Height*

Finally, we consider the sensitivity of the average impact survival probability to the height of the branch. In these calculations, the distance of impact point from the trunk varied randomly between 25 cm and 300 cm and the radius of the branch varied randomly between 0.5 cm and 10 cm, and constrained the height of the branch to relatively narrow ranges (usually 100 cm). The calculated average impact survivability probabilities are plotted in Figure 4 and values are given in Table 5.

What is interesting here is that the survival probability drops off sharply for branches higher than 3 m, but beyond that there is only a very gradual decline (Figure 4 and Table 5). What this reflects is that below a certain distance the time between the branch impact and the ground impact is too small for the spin imparted to the dart to result in a significant angle of attack. Above a certain distance, however, there is sufficient time for this excessive angle of attack to manifest itself.

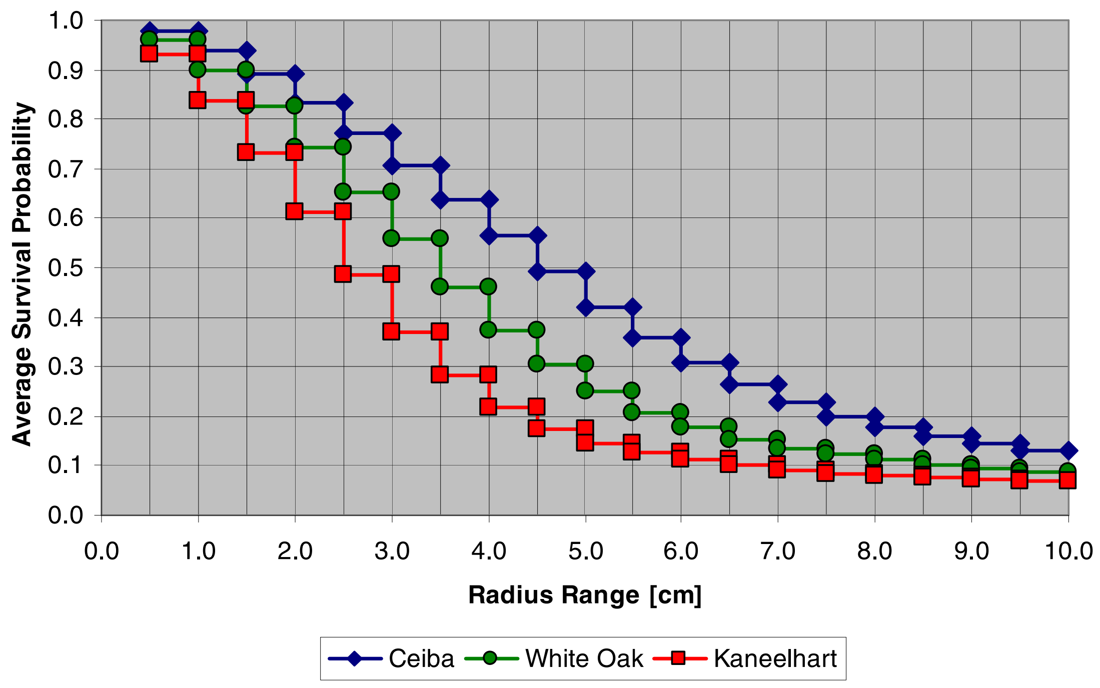

**Figure 3. Average Survival Probability, $P_{impact}$, for a Prototype Dart Design as a Function of Branch Radius, $r$**

**Table 4. Average Survival Probability, $P_{impact}$, as for a Prototype Dart Design as a Function of Branch Radius, $r$**

| | $P_{impact}$ | | |
|---|---|---|---|
| ***r* Range (cm)** | **Ceiba** | **White Oak** | **Kaneelhart** |
| 0.5 – 1.0 | 0.980 ± 0.002 | 0.961 ± 0.002 | 0.933 ± 0.002 |
| 1.0 – 1.5 | 0.939 | 0.899 | 0.839 |
| 1.5 – 2.0 | 0.890 | 0.824 | 0.730 |
| 2.0 – 2.5 | 0.833 | 0.741 | 0.612 |
| 2.5 – 3.0 | 0.771 | 0.653 | 0.484 |
| 3.0 – 3.5 | 0.705 | 0.559 | 0.368 |
| 3.5 – 4.0 | 0.637 | 0.461 | 0.282 |
| 4.0 – 4.5 | 0.566 | 0.374 | 0.218 |
| 4.5 – 5.0 | 0.493 | 0.305 | 0.175 |
| 5.0 – 5.5 | 0.421 | 0.249 | 0.146 |
| 5.5 – 6.0 | 0.360 | 0.207 | 0.126 |
| 6.0 – 6.5 | 0.308 | 0.176 | 0.111 |
| 6.5 – 7.0 | 0.264 | 0.153 | 0.100 |
| 7.0 – 7.5 | 0.228 | 0.135 | 0.091 |
| 7.5 – 8.0 | 0.199 | 0.122 | 0.084 |
| 8.0 – 8.5 | 0.177 | 0.111 | 0.079 |
| 8.5 – 9.0 | 0.158 | 0.102 | 0.074 |
| 9.0 – 9.5 | 0.143 | 0.095 | 0.071 |
| 9.5 – 10.0 | 0.131 | 0.089 | 0.068 |

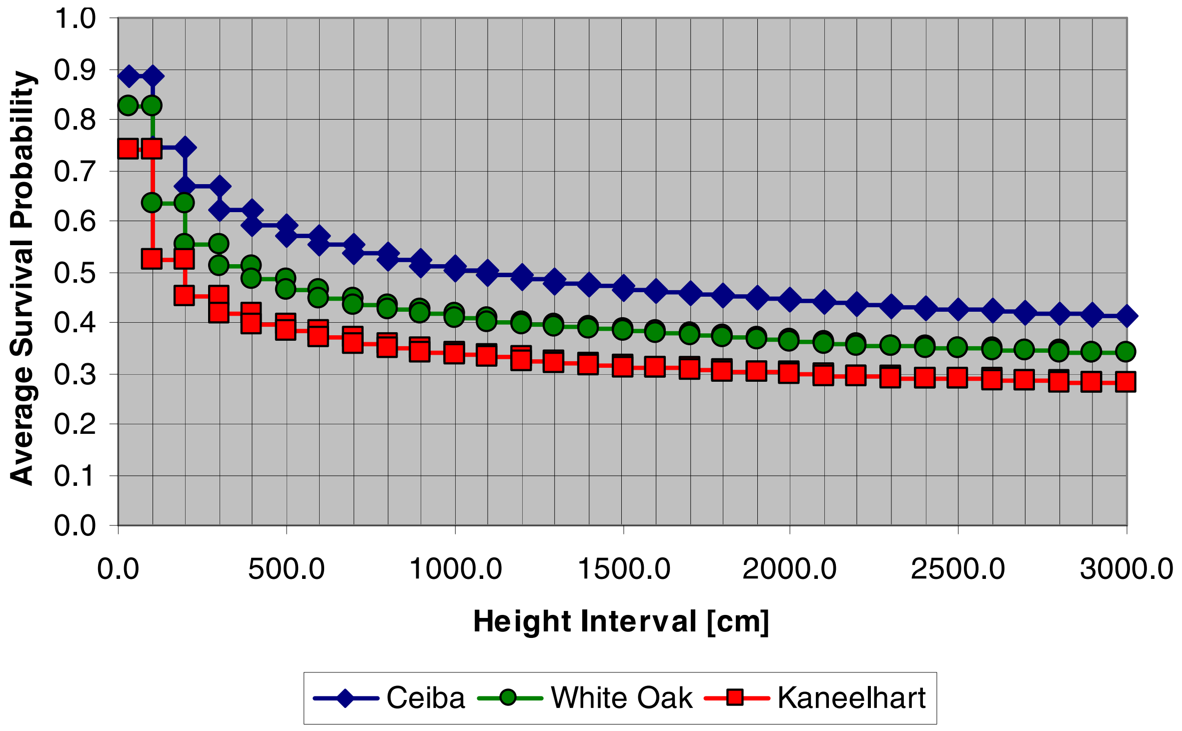

**Figure 4. Average Survival Probability, $P_{impact}$, for a Prototype Dart Design as a Function of Branch Height, *h***

## *4.7 Estimate of the Overall Branch-Impact Survival Probability*

In general, the single-branch impact probability at a site will depend on the species and growth history of the tree stand. However it is interesting to note that if the branch length varies randomly between 25 cm and 300 cm, the branch radius varies randomly between 0.5 cm and 10 cm, and the branch height varies randomly between 30 cm and 30 m, the average single-branch impact survival probability $P_{impact}$ ranges from 0.342 to 0.500 as the elastic modulus decreases (Table 6). It is therefore tempting to conclude that as a general rule of thumb, the probability of surviving the impact with a generic tree branch is on the order of 0.4. However, this number should be used with some caution.

## *4.8 Estimate of the Overall Site Survival Probability*

To estimate the probability of a dart surviving deployment at a forested site, we need the probability of impacting a tree canopy, $P_{canopy}$, and the probability of impacting a branch, $P_{branch}$. $P_{canopy}$ can be determined from the projected area of the canopy. For an estimate, we can use the conservative value $P_{canopy} = 1$, *i.e.*, dense forest.

As noted in section 2, the probability of impacting a branch, $P_{branch}$, is difficult to determine. It depends on the species of tree, the maturity of the tree, the growing conditions for the region (climate and soil), and other factors. For example, $P_{branch}$ has been determined be approximately 0.2 for some walnut trees, but can be much larger, especially for other species of trees. $P_{branch}$ can be estimated using remote sensing or using tree growth models (section 3). For an estimate, we use the conservative value $P_{branch} = 0.5$, *i.e.*, dense branches.

**Table 5. Survival Probability, $P_{impact}$, for a Prototype Dart Design as a Function of Branch Height, $h$**

| | $P_{impact}$ | | |
|---|---|---|---|
| ***h* Range (cm)** | **Ceiba** | **White Oak** | **Kaneelhart** |
| 30.0 – 100.0 | 0.886 ± 0.002 | 0.824 ± 0.002 | 0.742 ± 0.002 |
| 100.0 – 200.0 | 0.745 | 0.633 | 0.522 |
| 200.0 – 300.0 | 0.668 | 0.553 | 0.451 |
| 300.0 – 400.0 | 0.623 | 0.511 | 0.417 |
| 400.0 – 500.0 | 0.592 | 0.484 | 0.396 |
| 500.0 – 600.0 | 0.569 | 0.464 | 0.381 |
| 600.0 – 700.0 | 0.552 | 0.448 | 0.368 |
| 700.0 – 800.0 | 0.536 | 0.436 | 0.358 |
| 800.0 – 900.0 | 0.523 | 0.425 | 0.349 |
| 900.0 – 1000.0 | 0.511 | 0.417 | 0.343 |
| 1000.0 – 1100.0 | 0.501 | 0.409 | 0.336 |
| 1100.0 – 1200.0 | 0.492 | 0.402 | 0.330 |
| 1200.0 – 1300.0 | 0.484 | 0.397 | 0.325 |
| 1300.0 – 1400.0 | 0.478 | 0.391 | 0.320 |
| 1400.0 – 1500.0 | 0.471 | 0.386 | 0.316 |
| 1500.0 – 1600.0 | 0.465 | 0.382 | 0.312 |
| 1600.0 – 1700.0 | 0.460 | 0.377 | 0.310 |
| 1700.0 – 1800.0 | 0.454 | 0.373 | 0.306 |
| 1800.0 – 1900.0 | 0.450 | 0.369 | 0.303 |
| 1900.0 – 2000.0 | 0.445 | 0.365 | 0.300 |
| 2000.0 – 2100.0 | 0.440 | 0.362 | 0.298 |
| 2100.0 – 2200.0 | 0.437 | 0.359 | 0.295 |
| 2200.0 – 2300.0 | 0.434 | 0.355 | 0.293 |
| 2300.0 – 2400.0 | 0.430 | 0.352 | 0.291 |
| 2400.0 – 2500.0 | 0.427 | 0.350 | 0.290 |
| 2500.0 – 2600.0 | 0.424 | 0.348 | 0.288 |
| 2600.0 – 2700.0 | 0.420 | 0.346 | 0.286 |
| 2700.0 – 2800.0 | 0.418 | 0.343 | 0.284 |
| 2800.0 – 2900.0 | 0.415 | 0.341 | 0.283 |
| 2900.0 – 3000.0 | 0.413 | 0.339 | 0.281 |

**Table 6. Average Branch-Impact Survival Probability, $P_{impact}$, Over All Impact Conditions and Estimated Survival Probability, $P_{survival}$, for a Prototype Dart Design**

| | **Ceiba** | **White Oak** | **Kaneelhart** |
|---|---|---|---|
| $P_{impact}$ | 0.500 ± 0.002 | 0.413 ± 0.002 | 0.342 ± 0.002 |
| $P_{survival}$ | 0.750 | 0.706 | 0.671 |

Then using Eq. (1),

$$P = 1 - P_{canopy} P_{branch} \left(1 - P_{impact}\right) \approx 1 - 1 \times 0.5 \times \left(1 - 0.4\right) = 0.70 \, .$$

Therefore, under fairly conservative assumptions, more than two thirds of the darts deployed at a heavily forested site may reach the ground with a sufficiently small angle of attack to be able to embed there. Estimates of the overall survival probabilities for each of the woods are given in Table 6.

## 5 Calculations of $P_{impact}$ for the Current Dart Design

We repeated the study discussed in section 4 for the current sensor dart design (Figure 1). As discussed in section 4, the diversity of tree architecture makes it impossible to draw detailed conclusions about the survivability of the sensor dart without reference to the characteristics of a particular target site, but we can draw some general conclusions regarding the probability of survival of an impact with a single branch, $P_{impact}$, by considering calculations of $P_{impact}$ for a range of elastic moduli for green wood.

As in the previous study, the dart was assumed to survive impact if it reaches the ground with an angle of attack of less than a threshold value. The geometry, mass, and moment of inertia of the dart were supplied by Joseph E. Lucero. The study was conducted for the same three woods as in the previous study: ceiba, white oak, and kaneelhart (Table 1).

### 5.1 Modeling Assumptions

We assumed the dart was delivered with its design velocity of 75 m/s normal to the ground. Note also that for this study the dart terrabrakes were open during the impact with the branch.

As in the previous study, we assumed the threshold survival angle of attack for the prototype sensor dart is 5°, which is conservative value for dart survival: in drop tests at Tonopah Test Range darts impacted the ground at angles of attack significantly larger than this and they embedded in the ground and their internal components survived [27].

As in the previous study, we assumed a uniform probability distribution for each of the various branch parameters. The probability of survival was calculated using the code given in the appendix, which permits the height, diameter, and distance from the trunk to the impact point on the branch to be varied within some specified range. A Monte Carlo calculation was then performed, in each iteration of which a specific branch height, a diameter, and a distance from the trunk to the impact point were randomly selected. The average single branch impact survival probability $P_{impact}$ is calculated using Eq. (15).

As in the previous study, we assumed there is no correlation between the radius of the branch and the distance of the impact point from the branch. In general, the greater the distance between the trunk and the impact point, the smaller the branch radius will be.

Finally, as in the previous study, we assumed there is no correlation between the branch height and the branch radius. In general, branches at a greater height will tend to have a smaller radius at a given impact distance from the trunk than branches closer to the ground. Thus the values calculated for $P_{impact}$ are likely under-estimates of the probabilities.

### 5.2 Probability Convergence Study

To verify the convergence of the probability calculations, we varied the number of calculations per bin, $N_c$, from 100 to 1000 for the average probability of survival. A value $N_c = 250$ (corresponding to 62,500 Monte Carlo samples) was sufficient to achieve numerical convergence (Table 7).

**Table 7. Convergence of the Average Survival Probability, $P_{impact}$, for the Current Sensor Dart Design with the Number of Calculations per Bin, $N_c$**

|  | Average Survival Probability | | |
|---|---|---|---|
| $N_c$ | Ceiba | White Oak | Kaneelhart |
| 100 | 0.7160 | 0.6195 | 0.5268 |
| 200 | 0.7162 | 0.6186 | 0.5250 |
| 250 | 0.7177 | 0.6213 | 0.5271 |
| 300 | 0.7175 | 0.6190 | 0.5252 |
| 400 | 0.7183 | 0.6206 | 0.5272 |
| 500 | 0.7154 | 0.6184 | 0.5243 |
| 1000 | 0.7167 | 0.6198 | 0.5257 |

## 5.3 Angle of Attack Histories

Some typical angle-of-attack histories are shown in Figure 5. A dart trajectory calculation was terminated if the angle of attack reached the failure criterion, 5°. Darts that survive the impact with the tree branch, that is, whose angles of attack when they reached the ground were less than 5°, have angle-of-attack histories that are horizontal lines or lines with small slope. Only one of the darts in Figure 5, Dart 10, survived; most did not survive, that is, their angle of attack reached 5° before they reached the ground. The early history of the angle of attack indicates that the interaction of the dart with the branch can be complex (for example, the angle-of-attack history for Darts 1 and 11). Trajectories for the darts whose angle-of-attack histories are given in Figure 5 are shown in Figure 6; the Xs indicate the darts that did not survive. The color coding of the darts in Figure 5 and Figure 6 is consistent.

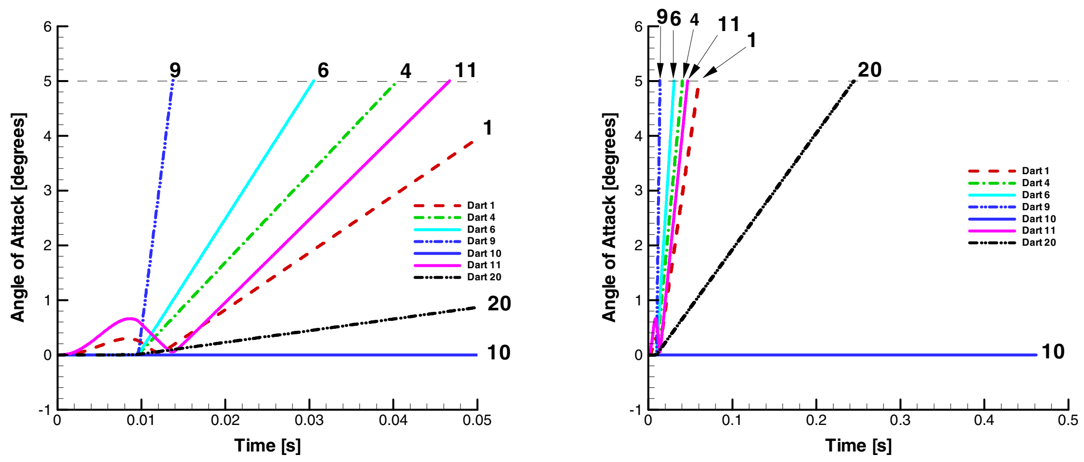

Figure 5. Angle-of-Attack Histories for Several Sensor Darts

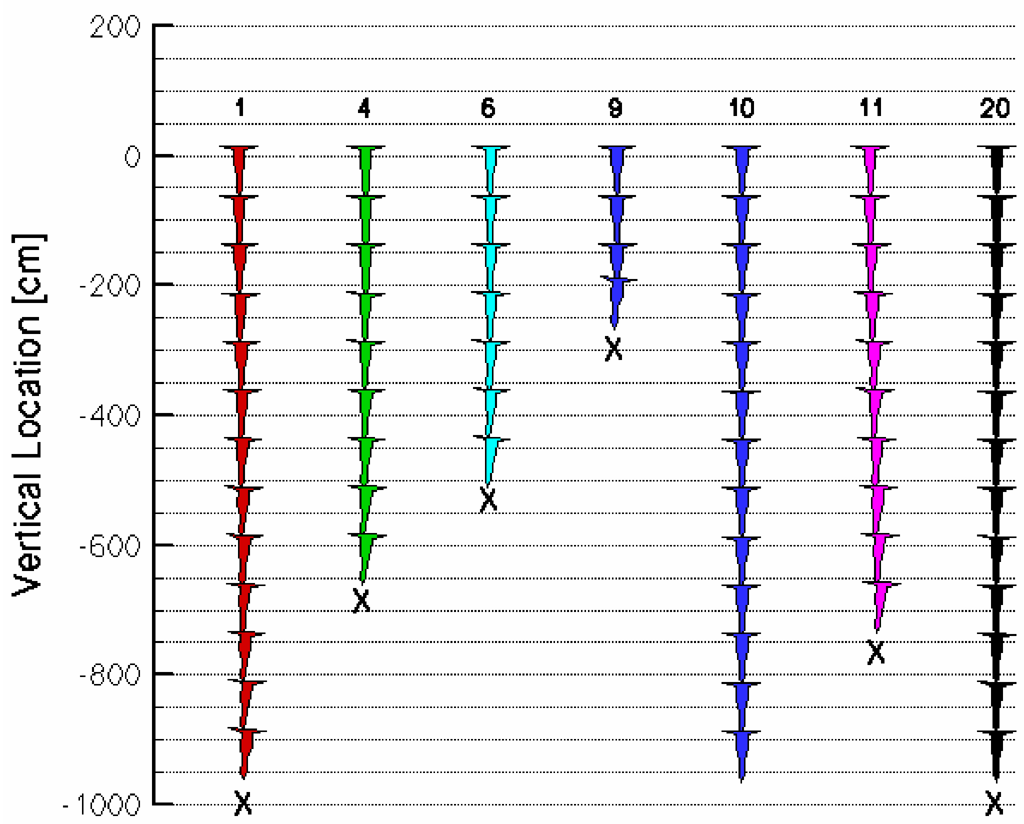

Figure 6. Trajectories for Several Sensor Darts

## 5.4 Survival Probability as a Function of Branch Impact Location

First we consider the sensitivity of the survival probability to how close the impact point on a branch is to the trunk of a tree[8]. In these calculations, the radius of the tree branch varied randomly between 0.5 cm and 10 cm and the height of the tree varied randomly between 30 cm and 30 m (approximately 100 ft). The distance between the trunk and the impact point was constrained to relatively narrow ranges of 25 cm or less. The calculated average impact survivability probabilities plotted in Figure 7 and the values are given in Table 8. Recall that the survival probability given here is only $P_{impact}$ ; it is not the overall survival probability for the dart deployed over a forest.

The survival probability decreases as the impact point gets closer to the trunk for each of the tree species. The physical explanation is obvious: the closer to the trunk the impact occurs, the stiffer the branch will be. Note that the radius of the extended terrabrakes is 26 cm (10.5 inches); an impact closer than this distance to the trunk would involve actually hitting the trunk, which would in general be unsurvivable for most trees. Not surprisingly, the probability of survival varies with the elastic modulus of the wood: The larger the elastic modulus of the wood is, the lower the average survival probability.

[8] The distance between the trunk and the impact point is statistically related to the length of the branch since distances within some range along a particular branch are sampled.

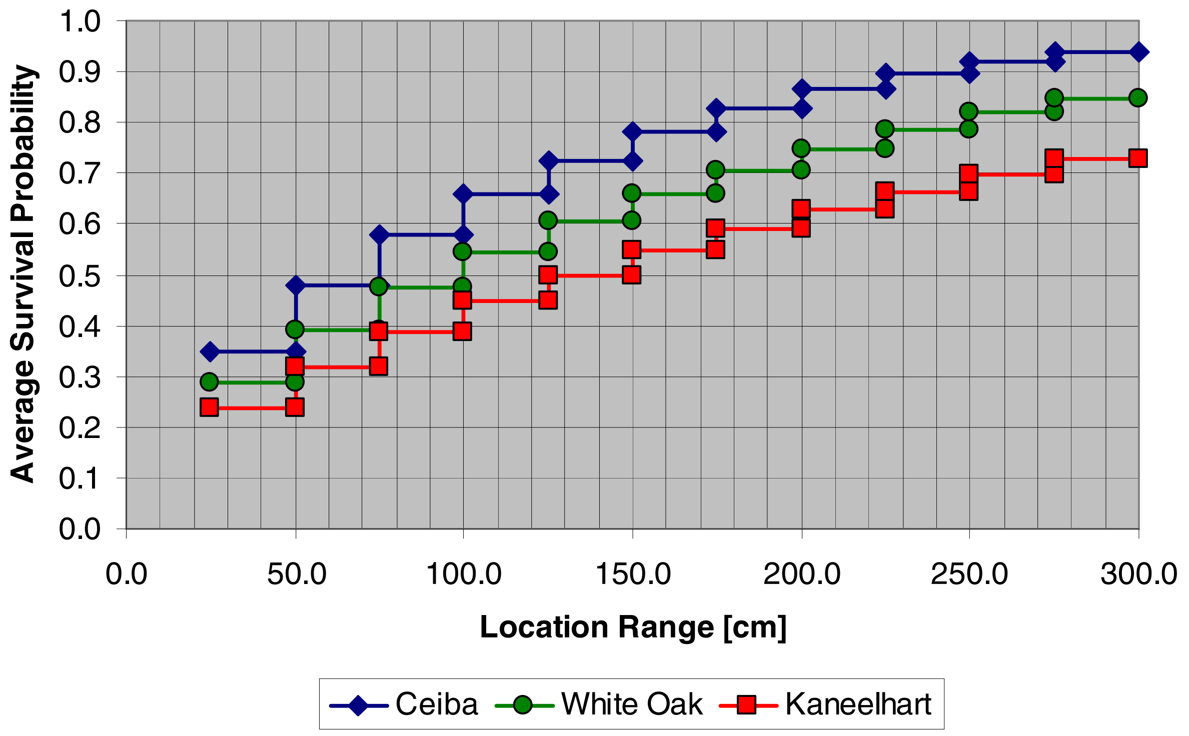

**Figure 7. Average Survival Probability, $P_{impact}$, for the Current Dart Design as a Function of Impact Location Measured from the Tree Trunk, $l$**

**Table 8. Average Survival Probability, $P_{impact}$, for the Current Dart Design as a Function of Impact Location Measured from the Tree Trunk, $l$**

| | $P_{impact}$ | | |
|---|---|---|---|
| **$l$ Range (cm)** | **Ceiba** | **White Oak** | **Kaneelhart** |
| 25.0 – 50.0 | 0.350 ± 0.002 | 0.287 ± 0.002 | 0.236 ± 0.002 |
| 50.0 – 75.0 | 0.480 | 0.391 | 0.320 |
| 75.0 – 100.0 | 0.578 | 0.475 | 0.387 |
| 100.0 – 125.0 | 0.658 | 0.544 | 0.447 |
| 125.0 – 150.0 | 0.725 | 0.606 | 0.499 |
| 150.0 – 175.0 | 0.781 | 0.658 | 0.546 |
| 175.0 – 200.0 | 0.827 | 0.707 | 0.589 |
| 200.0 – 225.0 | 0.865 | 0.749 | 0.629 |
| 225.0 – 250.0 | 0.895 | 0.786 | 0.664 |
| 250.0 – 275.0 | 0.919 | 0.818 | 0.697 |
| 275.0 – 300.0 | 0.938 | 0.846 | 0.728 |

## *5.5 Survival Probability as a Function of Branch Radius*

Next we consider the sensitivity of the branch-impact survival probability to the radius of the branch. In these calculations, the distance of the impact point from the trunk varied randomly between 25 cm and 300 cm and the height of the tree varied randomly between 30 cm and 30 m (approximately 100 ft). The radius of the branch was constrained to relatively narrow ranges of 0.5 cm. The calculated average impact survivability probabilities are plotted in Figure 8 and the values are given in Table 9.

The survival probability decreases rapidly as the branch radius increases (Figure 8 and Table 9). This is a consequence of the fact that the elastic restoring force of the branch is proportional to the area moment of inertia of the branch which in turn is proportional to the fourth power of the radius of the branch. As before, the higher the elastic modulus of the wood is, the lower the probability of survival.

## *5.6 Survival Probability as a Function of Branch Height*

Finally, we consider the sensitivity of the average impact-survival probability to the height of the branch. In these calculations, the distance of impact point from the trunk varied randomly between 25 cm and 300 cm and the radius of the branch varied randomly between 0.5 cm and 10 cm, and the height of the branch was constrained to relatively narrow ranges (usually 100 cm). The calculated average impact survivability probabilities are plotted in Figure 9 and the values are given in Table 10.

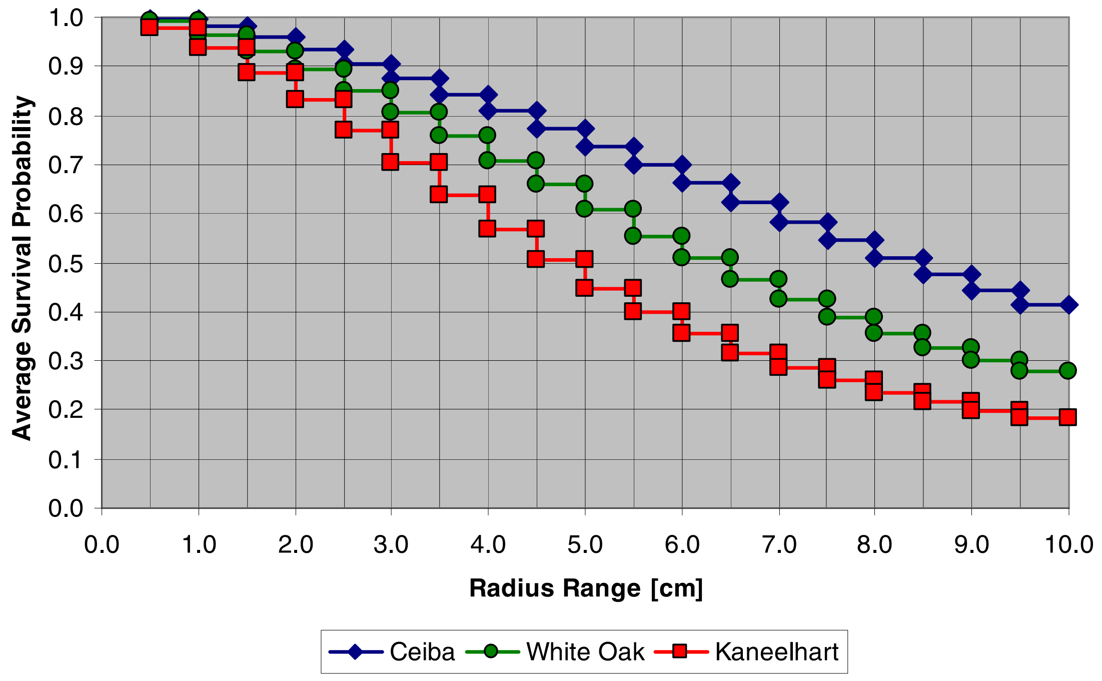

**Figure 8. Average Survival Probability, $P_{impact}$, for the Current Dart Design as a Function of Branch Radius, $r$**

**Table 9. Average Survival Probability, $P_{impact}$, as for the Current Dart Design as a Function of Branch Radius, $r$**

| | $P_{impact}$ | | |
|---|---|---|---|
| ***r* Range (cm)** | **Ceiba** | **White Oak** | **Kaneelhart** |
| 0.5 – 1.0 | 0.998 ± 0.002 | 0.991 ± 0.002 | 0.979 ± 0.002 |
| 1.0 – 1.5 | 0.982 | 0.963 | 0.936 |
| 1.5 – 2.0 | 0.958 | 0.929 | 0.887 |
| 2.0 – 2.5 | 0.933 | 0.892 | 0.830 |
| 2.5 – 3.0 | 0.906 | 0.850 | 0.768 |
| 3.0 – 3.5 | 0.876 | 0.805 | 0.703 |
| 3.5 – 4.0 | 0.844 | 0.757 | 0.638 |
| 4.0 – 4.5 | 0.809 | 0.708 | 0.569 |
| 4.5 – 5.0 | 0.773 | 0.659 | 0.505 |
| 5.0 – 5.5 | 0.737 | 0.607 | 0.449 |
| 5.5 – 6.0 | 0.698 | 0.555 | 0.399 |
| 6.0 – 6.5 | 0.662 | 0.508 | 0.355 |
| 6.5 – 7.0 | 0.623 | 0.464 | 0.317 |
| 7.0 – 7.5 | 0.583 | 0.424 | 0.285 |
| 7.5 – 8.0 | 0.545 | 0.388 | 0.259 |
| 8.0 – 8.5 | 0.509 | 0.356 | 0.235 |
| 8.5 – 9.0 | 0.476 | 0.326 | 0.216 |
| 9.0 – 9.5 | 0.444 | 0.299 | 0.199 |
| 9.5 – 10.0 | 0.415 | 0.277 | 0.184 |

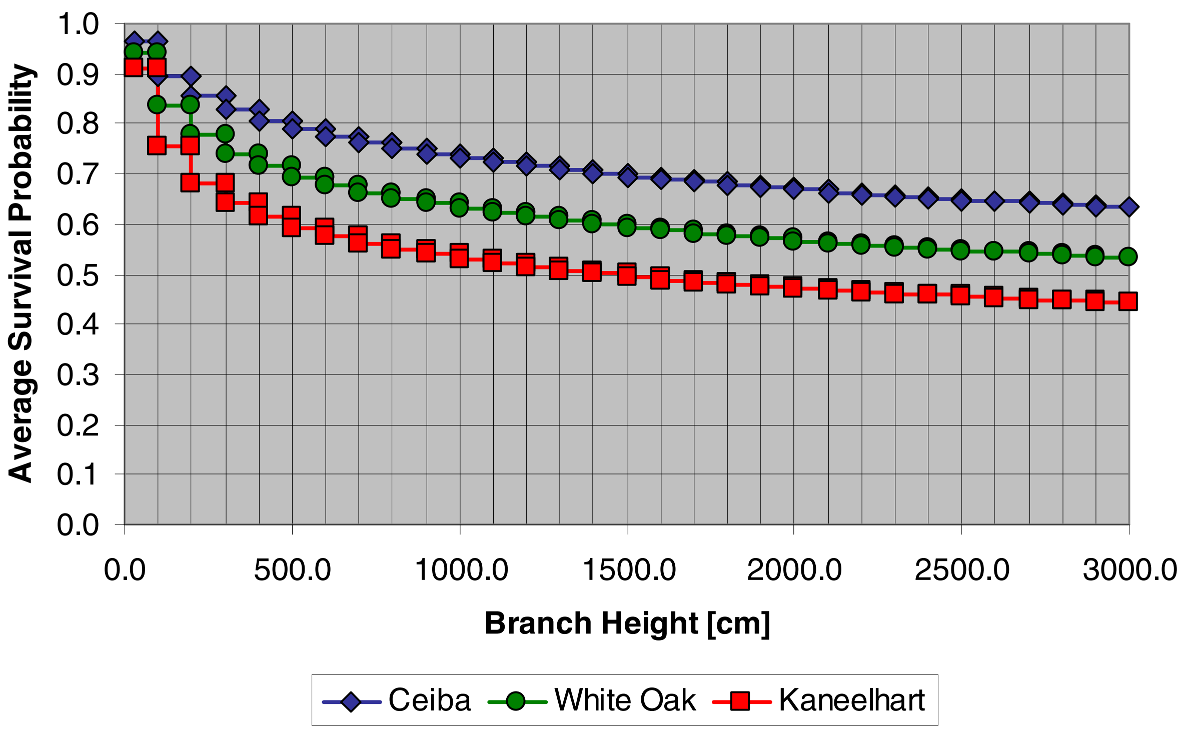

**Figure 9. Average Survival Probability, $P_{impact}$, for the Current Dart Design as a Function of Branch Height, $h$**

What is interesting here is that the survival probability drops off sharply for branches higher than 3 m, but beyond that there is only a very gradual decline (Figure 9 and Table 10). What this reflects is that below a certain distance the time between the branch impact and the ground impact is too small for the spin imparted to the dart to result in a significant angle of attack. Above a certain distance, however, there is sufficient time for this excessive pitch to manifest itself.

## 5.7 *Estimate of the Overall Branch-Impact Survival Probability*

In general, the single-branch impact probability at a site will depend on the species and growth history of the tree stand. However, it is interesting to note that if the branch length varies randomly between 25 cm and 300 cm, the branch radius varies randomly between 0.5 cm and 10 cm, and the branch height varies randomly between 30 cm and 30 m, the average single-branch impact survival probability $P_{impact}$ ranges from 0.527 to 0.718 as the elastic modulus decreases (Table 11). It is therefore tempting to conclude that as a general rule of thumb, the probability of surviving the impact with a tree branch in a generic tree is on the order of 0.6. However, this number should be used with some caution.

**Table 10. Average Survival Probability, $P_{impact}$, for the Current Dart Design as a Function of Branch Height, $h$**

| | $P_{impact}$ | | |
|---|---|---|---|
| ***h* Range (cm)** | **Ceiba** | **White Oak** | **Kaneelhart** |
| 30.0 – 100.0 | 0.964 ± 0.002 | 0.943 ± 0.002 | 0.909 ± 0.002 |
| 100.0 – 200.0 | 0.895 | 0.838 | 0.755 |
| 200.0 – 300.0 | 0.855 | 0.779 | 0.681 |
| 300.0 – 400.0 | 0.828 | 0.741 | 0.642 |
| 400.0 – 500.0 | 0.807 | 0.714 | 0.614 |
| 500.0 – 600.0 | 0.790 | 0.693 | 0.593 |
| 600.0 – 700.0 | 0.775 | 0.677 | 0.576 |
| 700.0 – 800.0 | 0.762 | 0.662 | 0.560 |
| 800.0 – 900.0 | 0.750 | 0.650 | 0.549 |
| 900.0 – 1000.0 | 0.739 | 0.640 | 0.539 |
| 1000.0 – 1100.0 | 0.730 | 0.631 | 0.530 |
| 1100.0 – 1200.0 | 0.722 | 0.622 | 0.521 |
| 1200.0 – 1300.0 | 0.714 | 0.614 | 0.513 |
| 1300.0 – 1400.0 | 0.706 | 0.606 | 0.507 |
| 1400.0 – 1500.0 | 0.699 | 0.599 | 0.501 |
| 1500.0 – 1600.0 | 0.694 | 0.593 | 0.494 |
| 1600.0 – 1700.0 | 0.689 | 0.587 | 0.488 |
| 1700.0 – 1800.0 | 0.683 | 0.581 | 0.483 |
| 1800.0 – 1900.0 | 0.679 | 0.576 | 0.478 |
| 1900.0 – 2000.0 | 0.673 | 0.571 | 0.473 |
| 2000.0 – 2100.0 | 0.668 | 0.565 | 0.470 |
| 2100.0 – 2200.0 | 0.663 | 0.561 | 0.466 |
| 2200.0 – 2300.0 | 0.660 | 0.557 | 0.464 |
| 2300.0 – 2400.0 | 0.656 | 0.553 | 0.461 |
| 2400.0 – 2500.0 | 0.651 | 0.549 | 0.458 |
| 2500.0 – 2600.0 | 0.648 | 0.546 | 0.455 |
| 2600.0 – 2700.0 | 0.647 | 0.544 | 0.452 |
| 2700.0 – 2800.0 | 0.641 | 0.540 | 0.449 |
| 2800.0 – 2900.0 | 0.638 | 0.537 | 0.446 |
| 2900.0 – 3000.0 | 0.635 | 0.534 | 0.444 |

## 5.8 Estimate of the Overall Site Survival Probability

To estimate the probability of a dart surviving deployment at a forested site, we need the probability of impacting a tree canopy, $P_{canopy}$, and the probability of impacting a branch, $P_{branch}$. $P_{canopy}$ can be determined from the projected area of the canopy. For an estimate, we can use the conservative value $P_{canopy} = 1$, *i.e.*, dense forest.

**Table 11. Average Branch-Impact Survival Probability, $P_{impact}$, Over All Impact Conditions and Estimated Survival Probability, $P_{survival}$, for the Current Dart Design**

| | **Ceiba** | **White Oak** | **Kaneelhart** |
|---|---|---|---|
| $P_{impact}$ | 0.718 ± 0.002 | 0.621 ± 0.002 | 0.527 ± 0.002 |
| $P_{survival}$ | 0.859 | 0.811 | 0.764 |

As noted in section 2, the probability of impacting a branch, $P_{branch}$, is difficult to determine. It depends on the species of tree, the maturity of the tree, the growing conditions for the region (climate and soil), and other factors. For example, $P_{branch}$ has been determined be approximately 0.2 for some walnut trees, but can be much larger, especially for other species of trees. $P_{branch}$ can be estimated using remote sensing or using tree growth models (section 3). For an estimate, we use the conservative value $P_{branch} = 0.5$, *i.e.*, dense branches.

Then using Eq. (1),

$$P = 1 - P_{canopy} P_{branch} \left(1 - P_{impact}\right) \approx 1 - 1 \times 0.5 \times \left(1 - 0.6\right) = 0.80 \,.$$

Therefore, under fairly conservative assumptions, approximately 80% of the darts deployed at a heavily forested site may reach the ground with a sufficiently small angle of attack to be able to embed there. Estimates of the overall survival probabilities for each of the woods are given in Table 11.

# 6 Summary and Conclusions

In this report we presented a methodology for computing the probability that the sensor dart for the "Near Real-Time Site Characterization for Assured HDBT Defeat" Grand-Challenge LDRD project will survive deployment over a forested region. The total probability can be decomposed into three approximately independent probabilities. Two of these account for the forest coverage and the branch density and the third accounts for the physics of an impact between the dart and a tree branch.

The probability that a dart survives an impact with a tree branch was determined from the effect of the impact on the dart trajectory. If a dart impacted the ground with an angle of attack less than a threshold angle, the dart was assumed to survive the branch impact; otherwise, it was assumed to not survive the impact. We developed a computer code for calculating the dart-tree impact survival probability based on this criterion. In the code, we modeled the sensor dart as a rigid body and a tree branch as a statically determinant, perfectly elastic beam with a cylindrical cross section. During an impact, the dart causes the branch to bend and the resulting elastic restoring force generates an impulse on the dart that changes the dart center of mass velocity and the angular velocity around the center of mass. The main approximations in the methodology were that the dart was treated as a non-deformable rigid body and aerodynamic forces were neglected. The rigid body approximation is reasonable because impacts capable of deforming or damaging the projectile are expected to induce a pitch and yaw to the rigid body that would result in destruction at ground impact irrespective of damage done during the branch impact. The neglect of aerodynamic forces is also considered acceptable given that any impact with a tree will occur close to the ground. We considered only single branch impacts.

We used a Monte Carlo scheme to calculate the probability of a sensor dart surviving an impact with a branch. Ranges of heights, radii, and lengths were defined for branches representative of a particular kind of tree. A specific branch height, radius, and impact point (less than or equal to the branch length) were chosen at random from within these ranges. The trajectory of the sensor dart was then calculated. If the dart experienced a rotation that exceeded a threshold angle of attack it was assumed to not survive the impact; otherwise it was assumed to have survived. This process was repeated on the order of 50,000 to 100,000 times with new, randomly chosen impact parameters for each calculation. The results of these calculations were then used to define a probability distribution in terms of the branch height, radius, and length.

The probabilities of encountering a tree and of impacting a tree branch depend on the nature of the forest and the trees themselves (species, maturity, growing conditions, *etc.*) and will be site-specific. We reviewed some methods for computing these probabilities in this report.

We conducted studies with two dart designs. One was an early prototype design in which the terrabrakes were closed during the branch impact. The second was the current dart design with the terrabrakes open during the branch impact.

As a rule of thumb, we observed that for reasonably generic trees and for a threshold angle of attack of 5°, which is conservative for dart survival based on actual drop tests of sensor darts, the probability of the dart reaching the ground with a sufficiently small angle of attack to be able to embed there after impact with a tree branch is on the order of 30% for the prototype dart design and 60% for the current dart design, although these values should be treated with some caution.

## Appendix: Listing of a Fortran 95 Computer Code for Computing $P_{impact}$

This appendix gives a listing of a Fortran 95 computer code for computing $P_{impact}$ based on a Monte Carlo analysis of the impact on a single branch of a tree. As presented here, the code computes a single survival probability based on randomly sampling different branch heights, radii, and lengths within a prescribed range of values. The numerical methodology used in this case is the one described in section 2, with minor but unimportant differences as documented in the code below.

The Monte Carlo parameters to be varied in this code are defined using parameter statements in the main program. The first three define bins over which the branch height, branch length (or equivalently distance of the impact from the trunk), and the branch radius are defined. These bins are only used for collecting statistics and do not affect the distribution of the parameters. The number of calculations per bin, $N_c$, determines how many points are randomly selected from each bin. All bins are sampled $N_c$ times over the course of a single calculation.

The range of values to be sampled for the branch height, impact point (branch length), and radius are given by the three parameters following the "number of calculations per bin" parameter. All values are in centimeters. The remaining parameters define the moment of inertia and mass of the dart, the maximum survival pitch at ground impact for the dart, and the Young's modulus for the tree.

This particular version of the code generates a simple average survival probability. However, the code has been written to collect data by bins in order to define a statistical probability distribution if desired. While the bin data is collected in this version, the probability distribution itself is not printed out.

```
      program survival_prob
!
!        Program to compute the probability of survival of the impact of a sensor dart
!        with a single tree branch.
!
!        John K. Prentice
!        Consultant, Sci-Tac, Inc.
!        7 January 2004
!
      USE dart_dynamics_m
!
      implicit none
!
      integer, parameter :: LONGreal=selected_real_kind(15,90)
!
!        Number of branch heights per impact point
!
      integer, parameter :: max_height_bins = 3
!
!        Number of impact bins along the length of each branch
!
      integer, parameter :: max_bins_along_tree = 50
!
!        Number of branch radius bins
!
      integer, parameter :: max_branch_radius_bins = 4
!
!        Number of calculations per bin
!
      integer, parameter :: number_calcs_per_bin = 100
!
!        Branch height extremes (1 foot to 10 feet)
!
      real (kind=LONGreal), parameter :: hmin = 30.48_LONGreal, hmax = 304.8_LONGreal
!
!        Branch impact point extremes (1 foot to 2 feet)
!
      real (kind=LONGreal), parameter :: lmin = 30.48_LONGreal, lmax = 60.96_LONGreal
!
!        Branch radius extremes (1/2 inch to 2 inches)
!
      real (kind=LONGreal), parameter :: rmin = 1.27_LONGreal, rmax = 5.08_LONGreal
!
!        Principal moments of inertia for the dart in dynes/cm^2.  Value is from
!        Rob Gilchrist, SNL
!
      real (kind=LONGreal), dimension(3), parameter ::                                  &
                            I_b = (/2.29e6_LONGreal, 2.29e6_LONGreal, 1.64e5_LONGreal/)
!
!        Mass for the dart in gm.  Value is from Rob Gilchrist, SNL
!
      real (kind=LONGreal), parameter :: mass = 13607.8
!
!        Maximum survivable pitch for the dart in degrees (Paul Yarrington has suggested
!        this value is give or take 5 degrees)
!
      real (kind=LONGreal), parameter :: max_survival_pitch = 5.0_LONGreal

!
!        Young's modulus for the tree.  For oak it is 1.7e11 dynes/cm^2 and for teak it
!        is 1.3e11 dynes/cm^2
!
      real (kind=LONGreal), parameter :: E = 1.3e11_LONGreal
!
!
```

```
!========== internal variables ============
!
!
      real (kind=LONGreal), dimension(3) :: branch_origin, branch_end, r_cm
      real (kind=LONGreal) :: branch_radius, dr, dz, dl, r0, z0, l0, rnow, znow, lnow,  &
                              random, impact_length, ypnt, total_survival_probability
      real (kind=LONGreal), dimension(max_height_bins) :: branch_height
      logical, dimension(max_height_bins) :: height_survival_array
      integer, dimension(max_bins_along_tree, max_branch_radius_bins,max_height_bins) ::&
                                                                           survival_matrix
      integer :: number_of_calculations, total_number_of_survivals, i, j, k, n
!
!         compute the branch impact point, branch radius, and height increments
!
      dr = (rmax-rmin) / real(max_branch_radius_bins-1, LONGreal)
      dl = (lmax-lmin) / real(max_bins_along_tree-1, LONGreal)
      dz = (hmax-hmin) / real(max_height_bins-1,LONGreal)
!
!         position the tree so that the trunk is along the positive z axis and branch is
!         along the positive x axis.  Because of assumptions made in the tree_dynamics
!         routine, this will require that the initial position of the dart be in the
!         x>0 and y<0 quarter-space.
!
      branch_origin(:) = (/0.0_LONGreal, 0.0_LONGreal, 0.0_LONGreal/)
      branch_end(:) = (/0.0_LONGreal, -lmax, 0.0_LONGreal/)
!
!         loop over all the various bins and compute the survivability
!
      survival_matrix(:,:,:) = 0
!
      do i = 1, max_bins_along_tree
          lnow = lmin + dl * real(i-1, LONGreal)
          l0 = lnow - 0.5_LONGreal * dl
!
          do j = 1, max_branch_radius_bins
              rnow = rmin + dr * real(j-1, LONGreal)
              r0 = rnow - 0.5_LONGreal * dr
!
              do n = 1, number_calcs_per_bin
!
!         determine the branch height, radius, and the point of impact along its length
!
                  call random_number (random)
                  branch_radius = max(0.1, r0 + random * dr)
                  call random_number (random)
                  impact_length = max(6.99_LONGreal, l0 + random * dl)
                  do k = 1, max_height_bins
                      znow = hmin + dz * real(k-1, LONGreal)
                      z0 = max(1.1_LONGreal*branch_radius, znow - 0.5_LONGreal * dz)
                      call random_number (random)
                      branch_height(k) = max(2.0_LONGreal*branch_radius, z0+random * dz)
                  end do
!
!         determine the center of mass coordinate for the dart for this iteration
!
                  call random_number (random)
                  r_cm(1) = 0.99_LONGreal * 6.99_LONGreal * random
                  r_cm(2) = -impact_length
                  r_cm(3) = 23.88_LONGreal
!
!       compute the survivability of the dart for this impact point and all heights
!
                  call dart_dynamics (branch_origin, branch_end, branch_radius, E, I_b, &
                                  r_cm, mass, max_survival_pitch, branch_height,        &
                                  height_survival_array)
                  do k = 1, max_height_bins
                      if (height_survival_array(k)) then
                          survival_matrix(i,j,k) = survival_matrix(i,j,k) +1
                      end if
                  end do
!
```

```
            end do
        end do
    end do
!
!       compute statistics
!
    number_of_calculations=max_height_bins*max_bins_along_tree*max_branch_radius_bins*&
                                                                number_calcs_per_bin
    total_number_of_survivals = 0
    do i = 1, max_bins_along_tree
        do j = 1, max_branch_radius_bins
            do k = 1, max_height_bins
                total_number_of_survivals = total_number_of_survivals +               &
                                                              survival_matrix(i,j,k)
            end do
        end do
    end do
    total_survival_probability = real(total_number_of_survivals, LONGreal) /          &
                                                  real(number_of_calculations, LONGreal)
!
!       output statistics
!
    print *,"Total number of calculations = ", number_of_calculations
    print *,"Total number of survivals = "   , total_number_of_survivals
    print *,"Total survival probability = "  , total_survival_probability

    end program survival_prob

module dart_dynamics_m
!
!       Calculate the motion of the sensor dart during the impact event
!
!       John K. Prentice
!       Consultant. Sci-Tac, Inc.
!       5 January 2004
!


USE dart_axes_m
USE tree_dynamics_m

implicit none

private
public :: dart_dynamics

contains
    subroutine dart_dynamics (branch_origin, branch_end, branch_radius, E, I_b, r_cm, &
                              mass, max_survival_pitch, branch_height, survival)
!
    integer, parameter :: LONGreal=selected_real_kind(15,90)
!
!=========== formal variables =============
!
    real (kind=LONGreal), dimension(:), intent(in) :: branch_height
    logical, dimension(:), intent(out) :: survival
    real (kind=LONGreal), intent(in) :: branch_radius, E, max_survival_pitch, mass
    real (kind=LONGreal), dimension(3), intent(in) :: branch_origin, branch_end, I_b
    real (kind=LONGreal), dimension(3), intent(inout) :: r_cm
!
!========== internal variables ============
!
    real (kind=LONGreal), dimension(3) :: torque, omega, omega_dot, f_cm, r_d0,       &
                                          r_d1, x_d_b, y_d_b, z_d_b, velocity, x_d,   &
                                          y_d, z_d, omega_inertial, omega_n, omega_nm1
    real (kind=LONGreal), dimension(3,3) :: alpha
    real (kind=LONGreal) :: t, dt, xmin, xmax, ymin, ymax, zmin, zmax, pitch
    integer :: i, j, k, number_of_branch_heights
    real (kind=LONGreal), dimension(:), allocatable :: final_pitch
    integer, parameter :: number_impact_time_steps = 250
    real (kind=LONGreal), parameter :: SMALL = epsilon(1.0_LONGreal)
```

```
!
!        determine the number of branch heights to evaluate and initialze some arrays
!
      number_of_branch_heights = size(branch_height)
      allocate (final_pitch(number_of_branch_heights))
      final_pitch(:) = -1.0_LONGreal
!
!        initialize velocity to 75 m/s and angular velocity to zero and the pitch to zero
!
      velocity(:) = (/0.0_LONGreal, 0.0_LONGreal, -7500.0_LONGreal/)
      omega_inertial(:) = 0.0_LONGreal
      x_d = (/1.0_LONGreal, 0.0_LONGreal, 0.0_LONGreal/)
      y_d = (/0.0_LONGreal, -1.0_LONGreal, 0.0_LONGreal/)
      z_d = (/0.0_LONGreal, 0.0_LONGreal, -1.0_LONGreal/)
      pitch = 0.0_LONGreal
!
!        compute the time step during the time any part of the dart is in the vacinity of
!        the tree branch
!
      dt = (58.42_LONGreal + 2.0_LONGreal*branch_radius)/abs(velocity(3))/              &
                                            real(number_impact_time_steps,LONGreal)
      t = 0.0_LONGreal
      omega_n(:)=0.0_LONGreal
      omega_nm1(:)=0.0_LONGreal
!
!        define the initial direction cosines for the body coordinate system relative to
!        the intertial system.  also define the z extremes for the dart in the interial
!        system
!
      alpha(1,1)=dot_product(x_d,(/1.0_LONGreal, 0.0_LONGreal, 0.0_LONGreal/))
      alpha(1,2)=dot_product(x_d,(/0.0_LONGreal, 1.0_LONGreal, 0.0_LONGreal/))
      alpha(1,3)=dot_product(x_d,(/0.0_LONGreal, 0.0_LONGreal, 1.0_LONGreal/))
      alpha(2,1)=dot_product(y_d,(/1.0_LONGreal, 0.0_LONGreal, 0.0_LONGreal/))
      alpha(2,2)=dot_product(y_d,(/0.0_LONGreal, 1.0_LONGreal, 0.0_LONGreal/))
      alpha(2,3)=dot_product(y_d,(/0.0_LONGreal, 0.0_LONGreal, 1.0_LONGreal/))
      alpha(3,1)=dot_product(z_d,(/1.0_LONGreal, 0.0_LONGreal, 0.0_LONGreal/))
      alpha(3,2)=dot_product(z_d,(/0.0_LONGreal, 1.0_LONGreal, 0.0_LONGreal/))
      alpha(3,3)=dot_product(z_d,(/0.0_LONGreal, 0.0_LONGreal, 1.0_LONGreal/))
!
      r_d0 = matmul (transpose(alpha),(/0.0_LONGreal,0.0_LONGreal,-34.54_LONGreal/))+r_cm
      r_d1 = matmul (transpose(alpha), (/0.0_LONGreal,0.0_LONGreal,23.88_LONGreal/))+r_cm
      zmin = min(r_d0(3),r_d1(3))
      zmax = max(r_d0(3),r_d1(3))
!
!        loop over time steps and integrate equations of motion accounting for impact
!        with the tree branch
!
all:  do
!
!        determine the center of mass force and the torque in the body coordinate system
!        due to the tree branch impact
!
         call tree_dynamics (branch_origin, branch_end, branch_radius, E, alpha, f_cm, &
                                                                       r_cm, torque)
!
!        transform the angular velocity from the inertial coordinate system to the body
!        coordinate system
!
         omega = matmul (alpha, omega_inertial)
!
!        compute the new angular velocity in the body coordinate system using Euler's
!        equations
!
         omega_dot(1) = (torque(1) - omega(2) * omega(3) * (I_b(2) - I_b(3))) / I_b(1)
         omega_dot(2) = (torque(2) - omega(1) * omega(3) * (I_b(3) - I_b(1))) / I_b(2)
         omega_dot(3) = (torque(3) - omega(1) * omega(2) * (I_b(1) - I_b(2))) / I_b(3)
!
         omega = omega_nm1 + 2*omega_dot*dt
         omega_nm1=omega_n
         omega_n = omega
!
```

```
!        compute the rotation of the dart due to this angular velocity in the body
!        coordinate system
!
          call dart_axes (omega, dt, x_d_b, y_d_b, z_d_b)
!
!        compute the new body axes in the inertial coordinate system
!
          x_d = matmul (transpose(alpha), x_d_b)
          y_d = matmul (transpose(alpha), y_d_b)
          z_d = matmul (transpose(alpha), z_d_b)
!
!        renormalize unit vectors
!
          x_d = x_d / sqrt(dot_product(x_d,x_d))
          y_d = y_d / sqrt(dot_product(y_d,y_d))
          z_d = z_d / sqrt(dot_product(z_d,z_d))
!
!        compute the new angular velocity in the inertial coordinate system
!
          omega_inertial = matmul (transpose(alpha), omega)
!
!        transform the cm forces from the body coordinate system to the inertial
!        coordinate system
!
          f_cm = matmul (transpose(alpha), f_cm)
!
!        compute the motion of the center of mass in the inertial coordinate system
!
          velocity = velocity + f_cm * dt / mass
          r_cm = r_cm + velocity*dt
!
!        define the direction cosines for the body coordinate system relative to the
!        inertial system
!
          alpha(1,1)=dot_product(x_d,(/1.0_LONGreal, 0.0_LONGreal, 0.0_LONGreal/))
          alpha(1,2)=dot_product(x_d,(/0.0_LONGreal, 1.0_LONGreal, 0.0_LONGreal/))
          alpha(1,3)=dot_product(x_d,(/0.0_LONGreal, 0.0_LONGreal, 1.0_LONGreal/))
          alpha(2,1)=dot_product(y_d,(/1.0_LONGreal, 0.0_LONGreal, 0.0_LONGreal/))
          alpha(2,2)=dot_product(y_d,(/0.0_LONGreal, 1.0_LONGreal, 0.0_LONGreal/))
          alpha(2,3)=dot_product(y_d,(/0.0_LONGreal, 0.0_LONGreal, 1.0_LONGreal/))
          alpha(3,1)=dot_product(z_d,(/1.0_LONGreal, 0.0_LONGreal, 0.0_LONGreal/))
          alpha(3,2)=dot_product(z_d,(/0.0_LONGreal, 1.0_LONGreal, 0.0_LONGreal/))
          alpha(3,3)=dot_product(z_d,(/0.0_LONGreal, 0.0_LONGreal, 1.0_LONGreal/))
!
          r_d0 = matmul(transpose(alpha),                                        &
                                   (/0.0_LONGreal,0.0_LONGreal,-34.54_LONGreal/))+r_cm
          r_d1 = matmul(transpose(alpha),                                        &
                                    (/0.0_LONGreal,0.0_LONGreal,23.88_LONGreal/))+r_cm
          zmin = min(r_d0(3),r_d1(3))
          zmax = max(r_d0(3),r_d1(3))
!
!         compute the new pitch
!
          pitch = acos(dot_product(z_d,(/0.0_LONGreal,0.0_LONGreal,-1.0_LONGreal/)))*   &
                                                180.0_LONGreal/3.1415925_LONGreal
!
!         advance the time step
!
          t = t + dt
!
!         if dart has hit the ground for a particular branch height, save the pitch.
!         If the dart has hit the ground from the highest branch, stop now.  Note, we
!         assume the branch_height array gives the heights in ascending order
!
          do j = 1, number_of_branch_heights
              if (zmin + branch_height(j) <= 0.0_LONGreal) then
                  if (final_pitch(j) <= -SMALL) then
                      final_pitch(j) = pitch
                  end if
                  if (j == number_of_branch_heights) exit all
              end if
```

```
!
!         if the pitch is greater than the maximum survivable pitch, stop now
!
              if (final_pitch(j) < -SMALL .AND. pitch > max_survival_pitch) then
                  final_pitch(j:number_of_branch_heights) = pitch
                  exit all
              end if
          end do
!
      end do all
!
!         determine survival
!
      where (final_pitch > max_survival_pitch)
          survival = .FALSE.
      elsewhere
          survival = .TRUE.
      endwhere
!
      deallocate (final_pitch)
!
      end subroutine dart_dynamics
!
end module dart_dynamics_m

module eulers_equations_m
!
!        Procedure to calculate the angular acceleration and nw angular velocity
!        of the sensor dart in the body coordinate system.  Input is the torque
!        in the body coordinate system, the angular velocity in the body
!        coordinate system, and the principal moments of inertia
!
!        John K. Prentice
!        Consultant. Sci-Tac, Inc.
!        4 January 2004
!

implicit none

private
public :: eulers_equations

contains
      subroutine eulers_equations (torque, omega, dt, omega_dot)
!
      integer, parameter :: LONGreal=selected_real_kind(15,90)
!
!=========== formal variables =============
!
      real (kind=LONGreal), dimension(3), intent(in) :: torque
      real (kind=LONGreal), intent(in) :: dt
      real (kind=LONGreal), dimension(3), intent(inout) :: omega
      real (kind=LONGreal), dimension(3), intent(out) :: omega_dot
!
!========== internal variables ============
!

!
      omega_dot(1) = (torque(1) - omega(2) * omega(3) * (I(2) - I(3))) / I(1)
      omega_dot(2) = (torque(2) - omega(1) * omega(3) * (I(3) - I(1))) / I(2)
      omega_dot(3) = (torque(3) - omega(1) * omega(2) * (I(1) - I(2))) / I(3)
!
      end subroutine eulers_equations
!
end module eulers_equations_m


module dart_axes_m
!
!        Procedure to calculate the rotation of the unit vectors defining the dart
!        relative to the body coordinate system.  See page 165 of Goldstein's
!        Classical Mechanics, 2nd edition, for the equation used to perform this
!        finite rotation around the angular velocity vector
!
!        John K. Prentice
!        Consultant. Sci-Tac, Inc.
!        5 January 2004
!

USE cross_product_m

implicit none

private ::
public :: dart_axes

contains
      subroutine dart_axes (omega, dt, x_d_b, y_d_b, z_d_b)
!
      integer, parameter :: LONGreal=selected_real_kind(15,90)
!
!=========== formal variables ============
!
      real (kind=LONGreal), dimension(3), intent(in) :: omega
      real (kind=LONGreal), intent(in) :: dt
      real (kind=LONGreal), dimension(3), intent(out) :: x_d_b, y_d_b, z_d_b
!
!========== internal variables ===========
!
      real (kind=LONGreal), dimension(3):: n, r, x_av, y_av, z_av
      real (kind=LONGreal) :: domega, phi, omega_mag, i, cos_domega, sin_domega, norm
      real (kind=LONGreal), dimension(3,3) :: alpha
      real (kind=LONGreal), parameter :: SMALL = epsilon(1.0_LONGreal)
!
      omega_mag = sqrt(dot_product(omega,omega))
      x_d_b = (/1.0_LONGreal, 0.0_LONGreal, 0.0_LONGreal/)
      y_d_b = (/0.0_LONGreal, 1.0_LONGreal, 0.0_LONGreal/)
      z_d_b = (/0.0_LONGreal, 0.0_LONGreal, 1.0_LONGreal/)
      domega = omega_mag * dt
      if (omega_mag > 0.0_LONGreal) then
!
!        compute the unit vectors for an angular velocity coordinate system
!        in the body coordinate system such that the angular velocity is along
!        the positive z axis of this new system
!
          z_av(:) = omega(:) / omega_mag
          norm = sqrt(dot_product(z_av, z_av))
          z_av(:) = z_av(:) / norm
          if (abs(z_av(1) - 1.0_LONGreal) <= SMALL) then
              x_av(:) = (/0.0_LONGreal, 1.0_LONGreal, 0.0_LONGreal/)
          else
              x_av(:) = cross_product (z_av,(/1.0_LONGreal, 0.0_LONGreal, 0.0_LONGreal/))
          end if
          norm = sqrt(dot_product(x_av, x_av))
          x_av(:) = x_av(:) / norm
          y_av(:) = cross_product (z_av, x_av)
          norm = sqrt(dot_product(y_av, y_av))
          y_av(:) = y_av(:) / norm
!
!        compute the rotation matrix between the body coordinate system and the
!        angular velocity system
!
          alpha(1,1)=dot_product(x_av,(/1.0_LONGreal, 0.0_LONGreal, 0.0_LONGreal/))
          alpha(1,2)=dot_product(x_av,(/0.0_LONGreal, 1.0_LONGreal, 0.0_LONGreal/))
          alpha(1,3)=dot_product(x_av,(/0.0_LONGreal, 0.0_LONGreal, 1.0_LONGreal/))
          alpha(2,1)=dot_product(y_av,(/1.0_LONGreal, 0.0_LONGreal, 0.0_LONGreal/))
          alpha(2,2)=dot_product(y_av,(/0.0_LONGreal, 1.0_LONGreal, 0.0_LONGreal/))
          alpha(2,3)=dot_product(y_av,(/0.0_LONGreal, 0.0_LONGreal, 1.0_LONGreal/))
          alpha(3,1)=dot_product(z_av,(/1.0_LONGreal, 0.0_LONGreal, 0.0_LONGreal/))
          alpha(3,2)=dot_product(z_av,(/0.0_LONGreal, 1.0_LONGreal, 0.0_LONGreal/))
          alpha(3,3)=dot_product(z_av,(/0.0_LONGreal, 0.0_LONGreal, 1.0_LONGreal/))
!
```

```
!        transform the dart coordinates into the angular velocity system
!
          x_d_b = matmul (alpha, x_d_b)
          y_d_b = matmul (alpha, y_d_b)
          z_d_b = matmul (alpha, z_d_b)
!
!        do the rotation of the dart around the angular velocity vector
!
          n = (/0.0_LONGreal, 0.0_LONGreal, 1.0_LONGreal/)
          cos_domega = cos(domega)
          sin_domega = sin(domega)
!
!        Note that we are interested here in counterclockwise rotations being
!        positive, hence we flip the sign on the cross-product below
!
          x_d_b = x_d_b*cos_domega + n*dot_product(n,x_d_b)*(1-cos_domega)-             &
                                                   cross_product(x_d_b,n)*sin_domega
          y_d_b = y_d_b*cos_domega + n*dot_product(n,y_d_b)*(1-cos_domega)-             &
                                                   cross_product(y_d_b,n)*sin_domega
          z_d_b = z_d_b*cos_domega + n*dot_product(n,z_d_b)*(1-cos_domega)-             &
                                                   cross_product(z_d_b,n)*sin_domega
!
!        rotate the new dart axes back into the body coordinate system
!
          x_d_b = matmul (transpose(alpha), x_d_b)
          y_d_b = matmul (transpose(alpha), y_d_b)
          z_d_b = matmul (transpose(alpha), z_d_b)
!
          norm = sqrt(dot_product(x_d_b, x_d_b))
          x_d_b(:) = x_d_b(:) / norm
          norm = sqrt(dot_product(y_d_b, y_d_b))
          y_d_b(:) = y_d_b(:) / norm
          norm = sqrt(dot_product(x_d_b, x_d_b))
          z_d_b(:) = z_d_b(:) / norm
      end if
!
      end subroutine dart_axes
!
end module dart_axes_m

module dart_radius_m
!
!        Compute the radius of the dart as a function of position
!        along the axis of symmetry (z=0 is the stern end of the dart).
!        Code automatically generated by Maple 9, then converted to F90
!        by hand.
!
!        znormal is a two-dimensional vector giving the (r,z) coordinates of
!        of the surface normal
!
!        John K. Prentice
!        Consultant. Sci-Tac, Inc.
!        4 January 2004
!

implicit none

private
public :: dart_radius

      integer, parameter, private :: LONGreal=selected_real_kind(15,90)
      real (kind=LONGreal), parameter, public :: mass = 13607.8
      real (kind=LONGreal), dimension(3), parameter, public ::                         &
                                I = (/2.29e6_LONGreal, 2.29e6_LONGreal, 1.64e5_LONGreal/)

contains
      subroutine dart_radius (z_in, radius, znormal)
!
!=========== formal variables =============
!
      real (kind=LONGreal), intent(in) :: z_in
      real (kind=LONGreal), intent(out) :: radius
      real (kind=LONGreal), dimension(2), intent(out) :: znormal
!
!========== internal variables ============
!
      real (kind=LONGreal) :: z, z_cm, length, rmin, rmax, r0, R, slope, magnitude
!
        z_cm = 34.54_LONGreal
        z = z_in + z_cm
!
!        point is above or below the dart
!
        if (z < 0.0_LONGreal .or. 58.42_LONGreal < z) then
          radius = -1.0_LONGreal
          znormal(:) = 0.0_LONGreal
!
!        point is inside lowest cylindrical section
!
        else if (z <= 6.64_LONGreal) then
          radius = 6.04_LONGreal
          znormal(:) = (/1.0_LONGreal, 0.0_LONGreal/)
!
!        point is in the fin section (also a cylinder)
!
        else if (z <= 16.99_LONGreal) then
          radius = 6.99_LONGreal
          znormal(:) = (/1.0_LONGreal, 0.0_LONGreal/)
!
!        point is in the conical section
!
        else if (z <= 32.66_LONGreal) then
          length = 13.81_LONGreal
          rmin = 3.81_LONGreal
          rmax = 6.99_LONGreal
          radius = rmax + (rmin - rmax) / length * (z - 16.99_LONGreal)
          slope = length/(rmax-rmin)
          znormal(:) = (/1.0_LONGreal, slope/)
          magnitude = sqrt(dot_product(znormal, znormal))
          znormal(:) = znormal(:) / magnitude
!
!        point is in the cylindrical region of the nose
!
        else if (z <= 46.47_LONGreal) then
          radius = 3.81_LONGreal
          znormal(:) = (/1.0_LONGreal, 0.0_LONGreal/)
!
!        point is in the ogive nose section
!
        else
          rmax = 3.81_LONGreal
          length = 11.95_LONGreal
          R = (rmax ** 2 + length ** 2) / (2.0_LONGreal*rmax)
          r0 = sqrt(R ** 2 - length ** 2)
          radius = sqrt(R ** 2 - (z - 46.47_LONGreal) ** 2) - r0
          znormal(1) = r0+radius
          znormal(2) = (z - 46.47_LONGreal)
          magnitude = sqrt(dot_product(znormal, znormal))
          znormal(:) = znormal(:) / magnitude
        end if
      end subroutine dart_radius
!
end module dart_radius_m

module tree_dynamics_m
!
!        Calculate the forces and torques exerted on the sensor dart by the branch
!
!        John K. Prentice
!        Consultant. Sci-Tac, Inc.
!        3 January 2004
!
```

```
USE dart_radius_m
USE cross_product_m

implicit none

private
public :: tree_dynamics

contains
      subroutine tree_dynamics (branch_origin, branch_end, branch_radius, E, alpha,     &
                                f_cm, r_cm, torque)
!
      integer, parameter :: LONGreal=selected_real_kind(15,90)
!
!=========== formal variables =============
!
      real (kind=LONGreal), intent(in) :: branch_radius, E
      real (kind=LONGreal), dimension(3), intent(in) :: branch_origin, branch_end, r_cm
      real (kind=LONGreal), dimension(3,3), intent(in) :: alpha
      real (kind=LONGreal), dimension(3), intent(out) :: f_cm, torque
!
!========== internal variables ============
!
      real (kind=LONGreal), parameter :: SMALL = epsilon(1.0_LONGreal)
      real (kind=LONGreal), parameter :: Pi = 3.1415926_LONGreal
      integer, parameter :: number_of_sample_points = 10
      real (kind=LONGreal), dimension(3) :: b0, b1, vector, normal,f_proj_vector,       &
                                     f_vector, deflection_normal, r_vector
      real (kind=LONGreal), dimension(2) :: rz_normal
      real (kind=LONGreal) :: dz, branch_length, snow, dradius, xpnt, ypnt, zpnt,       &
                              zmin, zmax, deflection, theta, magnitude, x0, x1, y0, y1, &
                              z0, z1, dmin, dmax, impact_distance, force_mag
      real (kind=LONGreal), dimension(number_of_sample_points) :: z_deflect,i_distance, &
                                                                  xp, yp, zp
      real (kind=LONGreal), dimension(3,number_of_sample_points) :: z_proj
      integer, dimension(1) :: i_array
      integer :: i
!
!         An important assumption made in this analysis is that the maximum survivable
!         pitch is small, only a few degrees.  This means that the axes of the body
!         system will always be nearly parallel to those of the inertial system (though
!         the z axis of the body system will in general be along the negative z axis
!         of the inertial system).  The importance of this assumption is that the side
!         of the dart where the branch is can be assumed to side that the branch has
!         always been on.  This is most critical in the determination of whether the
!         branch is in contact, we can ignore the possibility that somehow the branch
!         has ever been on the far side of the dart.
!
!         Associated with this assumption, we assume that the tree truck is in the x<0,
!         y<0 quarter-plane in the body coordinate system and that the unbent tree is
!         initially in the yz body coordinate plane.
!
!
!         Start by transforming the branch beginning and end into the body coordinate
!         system
!
      b0 = matmul (alpha, branch_origin - r_cm)
      b1 = matmul (alpha, branch_end - r_cm)
      branch_length = sqrt(dot_product(b0-b1,b0-b1))
!
!         If the dart is above or below the branch, zero the force and torque
!
      call dart_radius (min(b0(3),b1(3)), zmin, rz_normal)
      call dart_radius (max(b0(3),b1(3)), zmax, rz_normal)
      if (zmin < 0.0_LONGreal .AND. zmax < 0.0_LONGreal) then
          deflection = 0.0_LONGreal
      else
!
!         Otherwise compute the deflection of the tree and from that the force
!         and torque on the sensor dart.
```

```
!
!
!         Make sure that the tree trunk is in the x<0, y<0 quadrant.  Otherwise
!         all bets are off
!
!          if (b0(1) > 0.0_LONGreal .OR. b0(2) > 0.0_LONGreal) then
!              print *," "
!              print *,"Warning!!!!"
!              print *,"Tree trunk in the incorrect quadrant, abort"
!              print *,"r_cm=",r_cm
!              print *,"B0=",branch_origin
!              print *,"B1=",branch_end
!              print *,"branch radius=",branch_radius
!              print *,"b0=",b0
!              print *,"b1=",b1
!          end if
!
!         Case where the branch is at one z level
!
          if (abs(b0(3)-b1(3)) <= SMALL) then
              x0 = b0(1)
              y0 = b0(2)
              x1 = b1(1)
              y1 = b1(2)
              zpnt = b0(3)
              call dart_radius (zpnt, dradius, rz_normal)
              if (dradius <= 0.0_LONGreal) then
                  deflection = 0.0_LONGreal
!
!         Case where x coordinate of trunk is greater than or equal to -(dart radius)
!
              else if (x0 >= -dradius) then
                  xpnt = dradius * (dradius*x0/(x0**2+y0**2) -                      &
                                 sqrt((y0**2*(x0**2-dradius**2)+y0**4)/(x0**2+y0**2)**2))
                  ypnt = -sqrt(dradius**2 - xpnt**2)
                  impact_distance = sqrt((x0-xpnt)**2+(y0-ypnt)**2)
                  if (impact_distance >= branch_length) then
                      deflection = 0.0_LONGreal
                  else
                      vector=cross_product (b1-b0,(/xpnt-b0(1),ypnt-b0(2),0.0_LONGreal/))
                      if (vector(3) <= SMALL) then
                          deflection = 0.0_LONGreal
                      else
                          x1 = x0 + (x1-x0)/branch_length*impact_distance
                          y1 = y0 + (y1-y0)/branch_length*impact_distance
                          deflection=sqrt((x1-xpnt)**2+(y1-ypnt)**2)
                          theta = atan(abs(ypnt/xpnt))
                          normal(1) = rz_normal(1) * cos(theta)
                          normal(2) = rz_normal(1) * sin(theta)
                          normal(3) = -rz_normal(2)
                          magnitude = sqrt(dot_product(normal, normal))
                          normal(:) = normal(:) / magnitude
                          f_vector = (/x1-xpnt, y1-ypnt, 0.0_LONGreal/)
                          magnitude = sqrt(dot_product(f_vector, f_vector))
                          f_vector(:) = f_vector(:) / magnitude
                          f_proj_vector(:) = dot_product(f_vector, normal) * normal
                          magnitude = sqrt(dot_product(f_proj_vector, f_proj_vector))
                          f_proj_vector(:) = f_proj_vector(:) / magnitude
                      end if
                  end if
              else
!
!         Case where x coordinate of trunk is less than -(dart radius)
!
                  xpnt = dradius * (dradius*x0/(x0**2+y0**2) +                      &
                                 sqrt((y0**2*(x0**2-dradius**2)+y0**4)/(x0**2+y0**2)**2))
                  ypnt = sqrt(dradius**2 - xpnt**2)
                  impact_distance = sqrt((x0-xpnt)**2+(y0-ypnt)**2)
                  if (impact_distance >= branch_length) then
                      deflection = 0.0_LONGreal
                  else
```

```
                    vector=cross_product (b1-b0,(/xpnt-b0(1),ypnt-b0(2),0.0_LONGreal/))
                    if (vector(3) <= SMALL) then
                        deflection = 0.0_LONGreal
                    else
                        x1 = x0 + (x1-x0)/branch_length*impact_distance
                        y1 = y0 + (y1-y0)/branch_length*impact_distance
                        deflection=sqrt((x1-xpnt)**2+(y1-ypnt)**2)
                        theta = atan(abs(ypnt/xpnt))
                        normal(1) = rz_normal(1) * cos(theta)
                        normal(2) = -rz_normal(1) * sin(theta)
                        normal(3) = -rz_normal(2)
                        magnitude = sqrt(dot_product(normal, normal))
                        normal(:) = normal(:) / magnitude
                        f_vector = (/x1-xpnt, y1-ypnt, 0.0_LONGreal/)
                        magnitude = sqrt(dot_product(f_vector, f_vector))
                        f_vector(:) = f_vector(:) / magnitude
                        f_proj_vector(:) = dot_product(f_vector, normal) * normal
                        magnitude = sqrt(dot_product(f_proj_vector, f_proj_vector))
                        f_proj_vector(:) = f_proj_vector(:) / magnitude
                    end if
                end if
            endif
        else
!
!        Case where the branch is not at one z level
!
            dz = (b0(3) - b1(3)) / real(number_of_sample_points-1, LONGreal)
            x0 = b0(1)
            y0 = b0(2)
            x1 = b1(1)
            y1 = b1(2)
            z0 = b0(3)
            z1 = b1(3)
!
!        Case where x coordinate of trunk is greater than or equal to -(dart radius)
!
            if (x0 >= -dradius) then
                do i = 1, number_of_sample_points
                    zpnt = b0(3) + dz * real(i-1,LONGreal)
                    call dart_radius (zpnt, dradius, rz_normal)
                    if (dradius <= 0.0_LONGreal) then
                        z_deflect(i) = 0.0_LONGreal
                    else
                        xpnt = dradius * (dradius*x0/(x0**2+y0**2) -             &
                               sqrt((y0**2*(x0**2-dradius**2)+y0**4)/(x0**2+y0**2)**2))
                        ypnt = -sqrt(dradius**2 - xpnt**2)
                        impact_distance = sqrt((x0-xpnt)**2+(y0-ypnt)**2+(z0-zpnt)**2)
                        if (impact_distance >= branch_length) then
                            z_deflect(i) = 0.0_LONGreal
                        else
                            vector = cross_product(b1-b0,(/xpnt-b0(1),ypnt-b0(2),     &
                                                                      0.0_LONGreal/))
                            if (vector(3) <= SMALL) then
                                z_deflect(i) = 0.0_LONGreal
                            else
                                x1 = x0 + (x1-x0)/branch_length*impact_distance
                                y1 = y0 + (y1-y0)/branch_length*impact_distance
                                z1 = z0 + (z1-z0)/branch_length*impact_distance
                                z_deflect(i)=sqrt((x1-xpnt)**2+(y1-ypnt)**2+          &
                                                                        (z1-zpnt)**2)
                                theta = atan(abs(ypnt/xpnt))
                                normal(1) = rz_normal(1) * cos(theta)
                                normal(2) = rz_normal(1) * sin(theta)
                                normal(3) = -rz_normal(2)
                                magnitude = sqrt(dot_product(normal, normal))
                                normal(:) = normal(:) / magnitude
                                f_vector = (/x1-xpnt, y1-ypnt, z1-zpnt/)
                                magnitude = sqrt(dot_product(f_vector, f_vector))
                                f_vector(:) = f_vector(:) / magnitude
                                f_proj_vector(:) = dot_product(f_vector, normal)*normal
                                magnitude=                                            &
```

```
                                       sqrt(dot_product(f_proj_vector,f_proj_vector))
                              f_proj_vector(:) = f_proj_vector(:) / magnitude
                              z_proj(:,i) = f_proj_vector(:)
                              i_distance(i) = impact_distance
                              xp(i) = xpnt
                              yp(i) = ypnt
                              zp(i) = zpnt
                          end if
                      end if
                  end if
              end do
!
!        Case where x coordinate of trunk is less than -(dart radius)
!
          else
              do i = 1, number_of_sample_points
                  zpnt = b0(3) + dz * real(i-1,LONGreal)
                  call dart_radius (zpnt, dradius, rz_normal)
                  if (dradius <= 0.0_LONGreal) then
                      z_deflect(i) = 0.0_LONGreal
                  else
                      xpnt = dradius * (dradius*x0/(x0**2+y0**2) +              &
                             sqrt((y0**2*(x0**2-dradius**2)+y0**4)/(x0**2+y0**2)**2))
                      ypnt = -sqrt(dradius**2 - xpnt**2)
                      impact_distance = sqrt((x0-xpnt)**2+(y0-ypnt)**2+(z0-zpnt)**2)
                      if (impact_distance >= branch_length) then
                          z_deflect(i) = 0.0_LONGreal
                      else
                          vector = cross_product(b1-b0,(/xpnt-b0(1),ypnt-b0(2),     &
                                                              0.0_LONGreal/))
                          if (vector(3) <= SMALL) then
                              z_deflect(i) = 0.0_LONGreal
                          else
                              x1 = x0 + (x1-x0)/branch_length*impact_distance
                              y1 = y0 + (y1-y0)/branch_length*impact_distance
                              z1 = z0 + (z1-z0)/branch_length*impact_distance
                              z_deflect(i)=sqrt((x1-xpnt)**2+(y1-ypnt)**2+          &
                                                                    (z1-zpnt)**2)
                              theta = atan(abs(ypnt/xpnt))
                              normal(1) = rz_normal(1) * cos(theta)
                              normal(2) = -rz_normal(1) * sin(theta)
                              normal(3) = -rz_normal(2)
                              magnitude = sqrt(dot_product(normal, normal))
                              normal(:) = normal(:) / magnitude
                              f_vector = (/xpnt-x1, ypnt-y1, zpnt-z1/)
                              magnitude = sqrt(dot_product(f_vector, f_vector))
                              f_vector(:) = f_vector(:) / magnitude
                              f_proj_vector(:)=dot_product(f_vector, normal) * normal
                              magnitude=                                              &
                                       sqrt(dot_product(f_proj_vector,f_proj_vector))
                              f_proj_vector(:) = f_proj_vector(:) / magnitude
                              z_proj(:,i) = f_proj_vector(:)
                              i_distance(i) = impact_distance
                              xp(i) = xpnt
                              yp(i) = ypnt
                              zp(i) = zpnt
                          end if
                      end if
                  end if
              end do
          end if
!
          dmin = minval (z_deflect)
          dmax = maxval (z_deflect)
          if (max(dmin,dmax) <= 0.0_LONGreal) then
              deflection = 0.0_LONGreal
          else
              i_array = minloc (z_deflect, z_deflect > 0.0_LONGreal)
              i = i_array(1)
              deflection = z_deflect(i)
              normal(:) = z_proj(:,i)
```

```
                    impact_distance = i_distance(i)
                    f_proj_vector(:) = z_proj(:,i)
                    xpnt = xp(i)
                    ypnt = yp(i)
                    zpnt = zp(i)
               end if
            end if
      end if
!
!       compute force on the dart
!
      if (deflection > 0.0_LONGreal) then
          force_mag = 3.0_LONGreal*Pi*E*branch_radius**4*deflection/                      &
                                                   (4.0_LONGreal*impact_distance**3)
          f_cm(:) = f_proj_vector(:) * force_mag
          r_vector(1) = xpnt
          r_vector(2) = ypnt
          r_vector(3) = zpnt
          torque(:) = cross_product(r_vector, f_cm)
      else
          f_cm(:) = 0.0_LONGreal
          torque(:) = 0.0_LONGreal
      end if
!
      end subroutine tree_dynamics
!
end module tree_dynamics_m


module cross_product_m
!
!        Procedure to calculate vector cross-product
!
!        John K. Prentice
!        Consultant. Sci-Tac, Inc.
!        5 January 2004
!

implicit none

private ::
public :: cross_product

contains

      function cross_product (v1, v2) result(vout)
!
      integer, parameter :: LONGreal=selected_real_kind(15,90)
!
!=========== formal variables =============
!
      real (kind=LONGreal), dimension(3), intent(in) :: v1, v2
      real (kind=LONGreal), dimension(3) :: vout
!
!========== internal variables ============
!
      vout(1) = v1(2)*v2(3)-v1(3)*v2(2)
      vout(2) = v1(3)*v2(1)-v1(1)*v2(3)
      vout(3) = v1(1)*v2(2)-v1(2)*v2(1)
!
      end function cross_product
!
end module cross_product_m
```

## References

1. Landau, L.D. and Lifshitz, E.M, *Mechanics*, 3rd edition, Pergamon Press, 1976, p. 115.
2. Goldstein, Herbert, *Classical Mechanics*, 2nd edition, Addison –Wesley Publishing Company, 1980, pp. 143-148.
3. *ibid*, pp. 170-171.
4. Herbert L. Anderson, ed., *A Physicist's Desk Reference*, American Institute of Physics Press, 1989, p. 36.
5. Burton V. Barnes, Donald R. Zak, Shirley R. Denton, and Stephen H. Spurr, *Forest Ecology*, 4th edition, John Wiley and Sons, 1998, pp. 122-131.
6. P. B. Tomlinson, "Tree Architecture", *American Scientist* **71** (1983), pp. 142-149.
7. D. Barthelemy, C. Edelin, and F. Halle, "Canopy Architecture" in *Physiology of Trees*, A. S. Raghavendra, ed., John Wiley and Sons, 1991, pp. 2-20.
8. Tim Frevert, "Tree Architecture", Missouri Conservationist Online, September 1996, http://www.mdc.mo.gov/conmag/1996/09/20.html (11 April 2005).
9. —, "Canopy and Subcanopy Trees Arranged by Family", http://www.wlu.ca/~wwwbiol/bio305/Database/canopy.htm (11 April 2005).
10. —, "The Canopy", http://www.mongabay.com/0401.htm (11 April 2005).
11. S. N. Martens, S. L. Ustin, and J. Norman, "Measurement and Characterization of Tree Canopy Architecture*", International Journal of Remote Sensing* **12** (1991), pp. 1525-1545.
12. Deanna Weller, Robert Denham, Christian Witte, Celia Mackie., and Dave Smith, "Assessment and Monitoring of Foliage Projected Cover and Canopy Height Across Native Vegetation in Queensland, Australia, using Laser Profiler Data", *Can. J. Remote Sensing*, **29**(5) (2003), pp. 578-591.
13. David L. A. Gaveau and Ross A. Hill, "Quantifying Canopy Height Underestimation by Laser Pulse Penetration in Small-Footprint Airborne Laser Scanning Data", *Can. J. Remote Sensing*, **29**(5) (2003), pp. 650-657.
14. J. Z. Benson, "SYLVA, A Computer Model to Assess the Probability of Forest Penetration", Sandia National Laboratories Technical Report SAND89-8225, Livermore, CA, April 1989.
15. Jari Perttunen, Eero Nikinmaa, Martin J. Lechowicz, Risto Sievänen, and Christian Messier, "Application of the Functional-Structural Tree Model LIGNUM to Sugar Maple Saplings (*Acer saccharum* Marsh) Growing in Forest Gaps", *Annals of Botany* **88** (2001), pp. 471-481.
16. Harry T. Valentine, "Height Growth, Site Index, and Carbon Metabolism", *Silva Fennica* **31**(3) (1997), pp. 251-263.
17. Annikki Mäkelä, Petteri Vanninen, and Veli-Pekka Ikonen, "An Application of Process-Based Modeling to the Development of Branchiness in Scots Pine", *Silva Fennica* **31**(3) (1997), pp. 369-380.

18. Frédéric Boudon, Christopher Nouguier, and Christophe Godin, “Multiscale Geometric Representation of Heterogeneous Stands”, *Workshop Proceedings 2001-11, Third International Workshop on Functional-Structural Tree and Stand Models* (September 27-30, 2001, Van-Morin, QC).
19. C. Godin and Y. Caraglio, “A Multiscale Model for Plant Topological Structures”, *Journal of Theoretical Biology* **191** (1998), pp. 1-46.
20. C. Godin, E. Costes, and H. Sinoquet, “A Method for Describing Plant Architecture which Integrates Topology and Geometry”, *Annals of Botany* **84** (1999), pp. 343-357.
21. Przemyslaw Prusinkiewicz and Brendan Lane, “Integrating L-System Models of Plant Architecture and Plant Ecosysstems”, *Workshop Proceedings 2001-11, Third International Workshop on Functional-Structural Tree and Stand Models* (September 27-30, 2001, Van-Morin, QC).
22. Helge Dzierzon and Winfried Kurth, “Simulation of Tree Stands Using L-Systems and the Assessment of Results Shown by a Scots Pine Stand Example”, *Workshop Proceedings 2001-11, Third International Workshop on Functional-Structural Tree and Stand Models* (September 27-30, 2001, Van-Morin, QC).
23. Godin, E. Costes, and H. Sinoquet, “A Method for Describing Plant Architecture which Integrates Topology and Geometry”, *Annals of Botany* **84** (1999), pp. 343-357.
24. The AMAPmod Model, http://amap.cirad.fr (11 April 2005).
25. Hong-Ping Yan, Jean Francois Barczi, Philippe De Reffye, and Bao-Gang Hu, “Fast Algorithms of Plant Computation Based on Substructure Instances”, *Journal of WSCG’2002*, **10**(3), 2002, University of West Bohemia, Plzen, Czech Republic, 2002, pp.145-152, http://wscg.zcu.cz/wscg2002/Papers_2002/F37.pdf (11 April 2005).
26. Bloomenthal, Jules, “Modeling the Mighty Maple”, *SIGGRAPH ’85*, **19**(3) (1985), pp. 305-311.
27. Joseph E. Lucero, personal communication, January 2005.
28. Mencuccini, Maurizio; Grace, John; Fioravanti, Marco, “Biomechanical and Hydraulic Determinants of Tree Structure in Scots Pine: Anatomical Characteristics”, *Tree Physiology*, **17**(2), 105–113 (1997).
29. Green, David W.; Winandy, Jerrold E.; Kretschmann, David E., “Mechanical Properties of Wood” in *Wood Handbook: Wood as an Engineering Material*, USDA Forest Service, Forest Products Laboratory, Madison, WI, 1999. General technical report FPL; GTR-113: pp. 4.1-4.45. http://www.treesearch.fs.fed.us/pubs/viewpub.jsp?index=7149 (8 April 2005).
30. Chundnoff, M., *Tropical Timbers of the World*, Agriculture Handbook #607, U.S. Dept. of Agriculture, Forest Service, Forest Products Laboratory, Madison, WI, 1984, 464p.; see http://www.fpl.fs.fed.us/documnts/techline/iv-1.pdf (8 April 2005).
31. *ibid*, Kaneelhart (*Licaria* spp.), http://www2.fpl.fs.fed.us/TechSheets/Chudnoff/TropAmerican/htmlDocs_tropamerican/Licariaspp.html (8 April 2005).
32. *ibid*, Ceiba (*Ceiba pentandra* (Africa)), http://www2.fpl.fs.fed.us/TechSheets/Chudnoff/African/new_html_docs/Ceibapentandra.html (8 April 2005).

33. *ibid*, Ceiba (*Ceiba pentandra* (America)), http://www2.fpl.fs.fed.us/TechSheets/Chudnoff/African/new_html_docs/Ceibapentandra.html (8 April 2005).
34. White Oak, (*Quercus alba*), http://www2.fpl.fs.fed.us/TechSheets/HardwoodNA/pdf_files/quercusmet.pdf (13 April 2005).

## Distribution

| | |
|---|---|
| 1 | MS 0321<br>William J. Camp, 9200 |
| 1 | MS 0139<br>Paul Yarrington, 9902 |
| 3 | MS 0318<br>John K. Prentice, 9231 |
| 10 | MS 0316<br>David R. Gardner, 9233 |
| 1 | MS 0482<br>Ed R. Hoover, 2131 |
| 1 | MS 0482<br>John L. Sichler, 2131 |
| 1 | MS 0572<br>Eric P. Chael, 5736 |
| 1 | MS 0750<br>Gregory J. Elbring, 6116 |
| 1 | MS 0751<br>Laurence S. Costin, 6111 |
| 1 | MS 0859<br>K. Terry Stalker, 15432 |
| 1 | MS 0844<br>Robert J. Fogler, 15433 |
| 1 | MS 1160<br>Douglas A. Dederman, 15431 |
| 1 | MS 1160<br>Joseph E. Lucero, 15431 |
| 1 | MS 0417<br>Steven W. Hatch, 9743 |
| 1 | MS 0417<br>Steven M. Schafer, 9743 |
| 1 | MS 0123<br>Henry R. Westrich, 1011 |
| 1 | MS 0123<br>Keith Ortiz, 1011 |
| 1 | MS 1138<br>Central Technical Files, 8945-1 |
| 2 | MS 0899<br>Technical Library, 9616 |